Springer Proceedings in Advanced Robotics 2
Series Editors: Bruno Siciliano · Oussama Khatib

Antonio Bicchi
Wolfram Burgard *Editors*

Robotics Research

Volume 1

spar

Springer

Springer Proceedings in Advanced Robotics 2

Editors

Prof. Bruno Siciliano
Dipartimento di Ingegneria Elettrica
e Tecnologie dell'Informazione
Università degli Studi di Napoli
Federico II
Via Claudio 21, 80125 Napoli
Italy
E-mail: siciliano@unina.it

Prof. Oussama Khatib
Robotics Laboratory
Department of Computer Science
Stanford University
Stanford, CA 94305-9010
USA
E-mail: khatib@cs.stanford.edu

Antonio Bicchi · Wolfram Burgard
Editors

Robotics Research

Volume 1

Springer

Editors
Antonio Bicchi
Istituto Italiano di Tecnologia
Genoa
Italy

and

Research Center “E. Piaggio”
University of Pisa
Pisa
Italy

Wolfram Burgard
Institut für Informatik
Albert-Ludwigs-Universität Freiburg
Freiburg im Breisgau
Germany

ISSN 2511-1256 ISSN 2511-1264 (electronic)
Springer Proceedings in Advanced Robotics
ISBN 978-3-319-51531-1 ISBN 978-3-319-51532-8 (eBook)
DOI 10.1007/978-3-319-51532-8

Library of Congress Control Number: 2017930281

Printed on acid-free paper

This Springer imprint is published by Springer Nature
The registered company is Springer International Publishing AG
The registered company address is: Gewerbestrasse 11, 6330 Cham, Switzerland

Foreword

Robots! Robots on Mars and in oceans, in hospitals and homes, in factories and schools; robots fighting fires, making goods and products, saving time and lives. Robots today are making a considerable impact from industrial manufacturing to health care, transportation, and exploration of the deep space and sea. Tomorrow, robots will become pervasive and touch upon many aspects of modern life.

The *Springer Tracts in Advanced Robotics (STAR)* was launched in 2002 with the goal of bringing to the research community the latest advances in the robotics field on the basis of their significance and quality. During the latest fifteen years, the STAR series has featured publication of both monographs and edited collections. Among the latter, the proceedings of thematic symposia devoted to excellence in robotics research, such as ISRR, ISER, FSR, and WAFR, have been regularly included in STAR.

The expansion of our field as well as the emergence of new research areas has motivated us to enlarging the pool of proceedings to be published in STAR in the last few years. This has ultimately led us to launching a sister series in parallel to STAR. The *Springer Proceedings in Advanced Robotics (SPAR)* is dedicated to the timely dissemination of the latest research results presented in selected symposia and workshops.

The twelfth edition of "Robotics Research" edited by Antonio Bicchi and Wolfram Burgard in its 8-part volume is a collection of a broad range of topics in robotics. The content of these contributions provides a wide coverage of the current state of robotics research: the advances and challenges in its theoretical foundation and technology basis, and the developments in its traditional and new emerging areas of applications. The diversity, novelty, and span of the work unfolding in these areas reveal the field's increased maturity and expanded scope.

From its beautiful venue to its excellent program, the twelfth edition of ISRR culminates with this important reference on the current developments and new directions in the field of robotics—a true tribute to its contributors and organizers!

Stanford, USA
November 2016

Oussama Khatib
SPAR Editor

Preface

The 12th International Symposium of Robotics Research (ISRR 2015) was held from September 12–15, 2015, in Sestri Levante, Italy. The ISRR series on conferences began in 1983, and it is sponsored by the International Foundation of Robotics Research (IFRR), an independent organization comprised of top researchers around the world.

The goal of the ISRR is to bring together active, leading robotics researchers from academia, government, and industry, to assess and share their views and ideas about the state of the art of robotics and to discuss promising new avenues for future research exploration in the field of Robotics.

The choice of the location of ISRR 2015 reflects a tradition in ISRR, holding the conference in a beautiful place where the natural and cultural setting can inspire deeper and longer-sighted thoughts in the pauses of a very intense working program. Having the symposium in Italy was also meant to be suggestive of the ideal link between the most advanced robotics research with the ideas and dreams of the great engineers of the past. They, in particular those who are named "Renaissance Engineers," thought and dreamed of realizing intelligent machines, including robots, but could not build them. Nowadays, robotics technology can make this possible. Some ideas, like the openings toward human sciences and the concept of human-centered design, are as much valid now as they were at that time.

Special emphasis in ISRR 2015 was given to the emerging frontiers, such as the fields of flying robots, soft robotics and natural machine motion, hands and haptics, multi-robot systems, cognitive robotics and learning, humanoids and legged locomotion, robot planning and navigation, and knowledge-based robots.

The goal of the ISRR Symposia is to bring together active leading robotics researchers and pioneers from academia, government, and industry to assess and share their views and ideas about the state of the art of robotics and to discuss promising new avenues for future research. Papers representing authoritative reviews of established research areas as well as papers reporting on new areas and pioneering work were sought for presentation at the symposium. In addition to the open call, a well selected number of leading researchers have been solicited to contribute by personal invitation.

A Greatest Hits track was introduced in ISRR 2015. A small number of research papers which have been selected for the most prestigious awards in the last year have been invited for presentation. This offered a unique possibility to have a synoptic view of what the robotics community considered to be the best of robotics research and put it in a larger context.

During the four-day symposium, 49 papers were presented in a single track, to cover the broad research area of robotics; two forum sessions integrated the program by facilitating group discussions. Poster sessions were also held, in a very informal interactive style. The procedure to select the papers and the participants was very strict. A number of selected leading researchers were invited to be part of the program committee, providing overview talks and participating in the review process. In addition to an open call for contributions, researchers who had made significant new contributions to robotics were invited to submit papers to a competitive review process. All papers were reviewed by the Symposium Program Committee and the International Foundation of Robotics Research (IFRR, the symposium sponsor) for final acceptance.

The symposium included visits to several beautiful sites in the area, as well as at encouraging greater participant interaction, also by stimulating cultural discussions and reflection on robotics, its historical background, and its future challenges. It furthermore included a technical tour to the Instituto Italiano di Technologia where a large variety of leading edge robotics science and systems were presented to the participants.

This book collects the papers presented at the symposium, with authoritative introductions to each section by the chairs of the corresponding sessions.

The ISRR 2015 co-chairs/editors would like to thank Floriana Sardi, for their invaluable help in the organization of the program; Monica Vasco and Simona Ventriglia for their tireless secretarial work on local organization; Nick Dring for the management of the Web site; and Abhinav Valada for helping especially in the final assembly of this book.

Genoa/Pisa, Italy Antonio Bicchi
Freiburg im Breisgau, Germany Wolfram Burgard
August 2016

Contents

Part I
Flying Robots

Session Summary

The past ten years have seen an explosive growth in interest in unmanned aerial vehicles and the development of the new concept of *aerial robots*, that is, aerial vehicles able to autonomously interact with the world by means of exogenous sensors and virtual/physical interaction with the environment. There is now a recognized commercial market in a range of remote surveillance applications such as monitoring traffic congestion, environmental sensing, and regular inspection of infrastructure such as bridges, dam walls, power lines, commercial photography applications, and emergency response surveillance. The robotics research community is leading the development of enabling technology for these applications, as well as beginning to probe the boundaries of what may be possible in other more challenging applications such as aerial manipulation for construction/assembly tasks.

This session included a wide range of topics ranging from theoretical advances to more applied contributions and with relevance across the whole spectrum of flying robot applications. In our opinion, it is worth noting the emphasis on vision-based control of aerial vehicles in the talks presented. This is a natural consequence of the passive low-power, lightweight, and information-rich nature of the vision-sensing modality, making it ideal for light aerial vehicles. The broad scope of papers is also a natural consequence of the very diverse (and highly active) nature of research across the field of aerial robotics. For such a research field, the best that can be hoped for in a short session is a snapshot of activity in the field. This is indeed what the session "Flying Robots” at the International Symposium on Robotics Research was able to provide.

The first technical talk, by Siddall, Kennedy, and Kovac, presented a novel concept of a flying robot that can dive into water and then escape again to continue flying. The key technology in this design is a system for powerful, repeatable generation of thrust in order to escape the water. The paper proposed a system that used acetylene explosions in a 34-gram water jet thruster, which expels water collected from its environment as propellant. Miniaturization problems of combustible fuel control and storage were addressed by generating acetylene gas from solid calcium carbide, which is reacted with environmental water. The resulting system produced over 20N of thrust, sufficient to propel small robots into the air from water. The ability to move between air and water with miniature robots has application in environmental monitoring for distributed water sampling and monitoring of a variety of unstructured marine environments, such as coral reefs and coastal areas.

In the second talk, Ritz and D'Andrea considered the question of controlling a tail-sitter aerial vehicle. Such systems have highly nonlinear dynamics, and simple local control designs are insufficient to recover to hover from a large set of initial conditions. The proposed attitude control law was based on an optimal control problem, with the objective of correcting large attitude errors by turning primarily around the vehicles strongly actuated axis. Solutions for a set of initial attitudes are precomputed and stored in a lookup table. For each controller update, the optimal inputs are read from this table and applied to the system in an MPC-like manner. The resulting control was shown to be robust and highly effective in bringing a tail-sitter vehicle back into hover from any configuration.

In the third talk, Do, Carrillo-Arce, and Roumeliotis considered visual control of quadrotors through a known environment. They focused on the case where collection of images is already available and the desired path is represented as a graph of linked images. The quadrotor follows the desired path by iteratively determining the desired motion to the next reference frame, and then, a visual servoing controller is used to reduce the error between current and desired image. The accuracy and robustness of the algorithm were demonstrated by navigating two quadrotors along lengthy corridors and through tight spaces inside a building and in the presence of dynamic obstacles.

The quality of papers in the session was exceptional, and several of the authors were only able to give short talks in the oral session, although all authors provided poster sessions later that day.

Helbling, Fuller, and Wood continue the Harvard research program into insect-scale robotics. The inherent instability of such systems, exacerbated by the faster dynamics that result from increasing angular accelerations with decreasing scale, requires high bandwidth sensing to maintain stable flight. In this paper, they build on previous work by incorporating a sensor that is size- and power-compatible with the Harvard RoboBee and is capable of estimating the distance by measuring the time of flight of an infrared laser pulse. The goal of the sensor is to provide sensor data that can be used to regulate altitude of an insect like flying robot. This work on onboard altitude control represents the latest results in achieving autonomous control and visually guided flight for such small-scale flying robotic systems.

Alhinai, Braithwaite, Haas-Hegerm, McFarlane, and Kovac considered the problem of using aerial robots for construction tasks. They took a highly novel approach based on multimodality of the vehicles and tensile construction. They designed and implemented mechanical and electronic designs of two payload packages for attachment to nanoquadrotor robots with a total integrated mass of only 26 g per robot. The payloads enabled a team of autonomous nanoaerial vehicles to construct a multielement tensile structure between anchor points in an irregular environment, such as a natural woodland. In addition to the mechanism, they provided trajectory planning and control algorithms required to enable robust execution of the construction scheme. The work was undertaken for nanovehicles but provides a indication of how aerial robots can move to take key roles in construction.

Tan, Lohmiller, and Slotine considered the general problem of simultaneous localization and mapping (SLAM) for general 3D position measurements. The approach taken applies a combination of linear time varying (LTV) Kalman filtering and nonlinear contraction tools in a fashion which avoids linearized approximations altogether. By exploiting fictive synthetic measurements, the LTV Kalman observer avoids errors and approximations brought by the linearization process in the EKF SLAM. Conditioned on the robot position, the covariances between landmarks are decoupled, making the algorithm easily scalable.

Costante, Delmerico, Werlberger, Valigi, and Scaramuzza consider the question of vision-based localization for aerial robotic vehicles. Such systems rely heavily on highly textured images in order to achieve accurate pose estimation. The authors point out that most path-planning strategies use only geometric information and consequently can cause an aerial vehicle to fly away from key visual information. Their path planner exploits the scenes' visual appearance (i.e., the photometric information) in combination with its 3D geometry. The approach is demonstrated with the real and simulated micro-aerial vehicles (MAVs) that perform perception-aware path planning in real time during exploration and demonstrate significantly reduced pose uncertainty over trajectories planned without considering the perception of the robot.

Huang and Leonard are also concerned with the question of visual navigation. In this paper, they introduce a new optimal state constraint for visual inertial navigation systems. Their proposed solution uses a tightly coupled visual inertial sensor fusion algorithm undertaken on a sliding window of poses. The key novelty is a novel measurement model that utilizes all feature measurements available within the sliding window and derives probabilistically optimal constraints between poses while without estimating these features as part of the state vector. For each sliding window, they perform structure and motion using only the available camera measurements and subsequently marginalize out the structure (features) to obtain the optimal motion constraints that will be used in the EKF update. The resulting motion estimation is validated in real-world experiments.

High-Power Propulsion Strategies for Aquatic Take-off in Robotics

Robert Siddall, Grant Kennedy and Mirko Kovac

1 Introduction

1.1 Aquatic Micro Air Vehicles

Distributed water sampling by small unmanned aerial vehicles has applications in disaster relief, oceanography, and agriculture, in which the monitoring of water contaminants is both time and resource intensive. In remote or dangerous marine environments, aerial vehicles can provide samples to a base station over a broad area, and can respond more rapidly than aquatic or terrestrial systems, increasing the spatial range of data that can be collected. We have proposed that implementing aquatic locomotion capability onto a fixed wing flying vehicle offers a low cost, versatile solution [16] (Fig. 1).

We are developing a small fixed wing aquatic Micro Air Vehicle (AquaMAV) able to dive directly into the water from the air and subsequently retaking flight. A fixed wing vehicle provides greater range and speed than hovering vehicles, particularly at the small scale, and the plunge diving approach reduces the need for accurate control, which allows for platforms to be produced at lower cost and operated in larger numbers. One of the most significant challenges in the implementation of a plunge diving miniature aerial robot is the provision of sufficient power for transition to flight from water. The provision of highly buoyant pontoons for a floatplane take off greatly

R. Siddall and G. Kennedy contributed equally to this work.

R. Siddall (✉) · G. Kennedy · M. Kovac
Imperial College London, London, UK
e-mail: r.siddall13@imperial.ac.uk

G. Kennedy
e-mail: grant.kennedy09@imperial.ac.uk

M. Kovac
e-mail: m.kovac@imperial.ac.uk

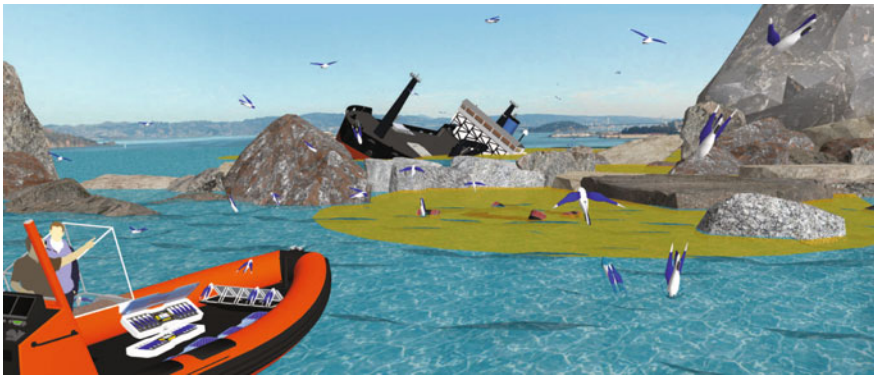


Fig. 1 Plunge diving AquaMAVs map a marine accident, retaking flight in a cluttered aquatic environment using powerful bursts of water jet thrust (adapted from [16])

inhibits aquatic locomotion, and makes diving from flight extremely difficult. A minimally buoyant vehicle must instead be able to produce significant thrust in order to propel itself out of the water. We propose that the best way to realise an AquaMAV is with the use of a powerful, impulsive thruster that can be easily integrated onto a flying platform.

Several projects have already demonstrated the efficacy of water sampling with UAVs using larger rotary wing vehicles [14, 15], by taking samples whilst hovering near the water surface. This approach relies on accurate sensing and control to maintain position while a sample probe is lowered, and has a limited range. Several large unmanned seaplanes are currently in operation [12, 16], experimental studies have shown the potential of an aerial-aquatic robot that is propelled by adaptable flapping wings [5, 9], or able to plunge dive into water [7]. Other work has demonstrated the efficacy of jumpgliding locomotion in terrestrial robots [3, 21, 23], and fixed wing Micro Air Vehicles (MAVs) have been implemented with terrestrial mobility [6]. Amphibious robots have been implemented in many forms [4, 8], but these robots are not able to cross large, sheer obstacles, and generally can only exit the water on gentle inclines. Such robots would have limited use in situations with large amounts of debris separating bodies of water (e.g. a flooded town). Despite the recent research effort in this emerging topic of investigation, to the best of the authors' knowledge no Aquatic MAV (AquaMAV) has been realised to date.

1.2 High Power Density Actuation

The challenge of equipping robots with large power reserves for rapid, impulsive movements is a recurring problem in robotics. This is often addressed with the use of elastic energy storage, compressed gas, or combustible fuels, all of which can

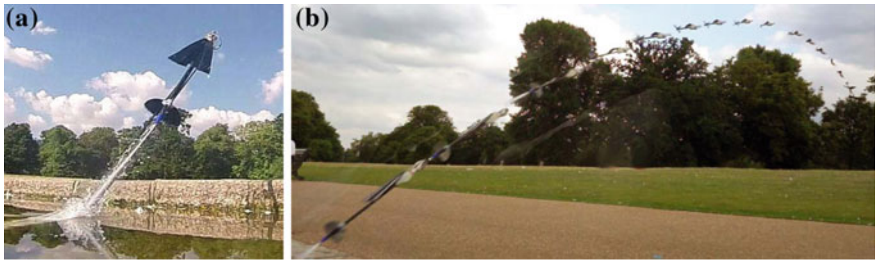


Fig. 2 **a** A jet propelled jumpgliding robot launching out of water using a CO_2 powered water jet thruster. **b** A timelapse image of the robot's trajectory, with wings deployed in the final still [17]

release large amounts of energy in a short space of time. A recent example is the 'Sandflea' robot produced by Boston Dynamics, a 5 kg wheeled vehicle that uses a CO_2 powered piston to jump 9 m into the air. Siddall and Kovač [18] created a 100 g jumpgliding robot with deployable wings, that was able to leap out of the water into flight using a miniature CO_2 tank to propel water through a nozzle (Fig. 2). However, this prototype must be recharged between actuations.

Combustion offers an alternative means of creating high pressures, without the need for a pre-pressurised tank and release mechanism. Churaman et al. [2] created robots weighing only 314 mg which were able to jump over 8 cm vertically using explosive actuators. A larger robot was built by researchers at Sandia Laboratories, which used combustion to power piston driven jumping, achieving 4,000 2 m jumps from 20 g of fuel [22]. More recent work has focussed on the use of combustion to power soft terrestrial jumping robots; [10, 20] have demonstrated the use of explosive hydrocarbon gas for locomotion in small soft robots. These systems use pressurised liquid reservoirs of butane gas as fuel, metered into a combustion chamber by electronic valves, and ignited by a spark.

However, the provision of multiple liquid fuel tanks and flow control apparatus is a significant mass and complexity penalty for a miniature flyable thruster. In this paper, we present a method for producing water jet thrust explosively, in a system that weighs only 34 g, using a solid fuel reserve to produce combustible acetylene gas, which is ignited in a valveless combustion chamber.

2 Solid Reactants as a Combustion Gas Source

Large volumes of fuel gas can be stored in a small space as a liquid under pressure. However, this means that the storage and regulation systems must be able to sustain the large pressures required, which makes components considerably heavier, and the provision of a pressurised container of combustible gas is a hazard in and of itself. If the combustible fuel is instead produced by the reaction of two separately stable components, high pressures can be avoided and the fuel storage and dispensing systems can be greatly simplified. In the particular case of an aquatic robot, water can be used as a reactant, exploiting the robot's environment to reduce system mass.

The gas production efficiencies for several water-reacting solid fuels are summarised in Table 1, with fuel required calculated based on a stoichiometric pre-combustion mix with air at standard temperature and pressure (STP). The energy content of the gases given off (again for combustion in air at STP) are given in Table 2, with the butane used by [10, 20] included for reference. Energy density is based on the fuel's high heating value (HHV, values taken from [11]), which includes energy recovery from the condensation of steam given off by the reaction, to consider work extraction from the products down to ambient temperature.

Table 1 Water-reactive solid fuel production efficiency, the combustible gas product is highlighted in **bold**

Fuel	Reaction	mg fuel per cm^3 combustor volume	μl fuel per cm^3 combustor volume
Lithium (Li)	$2Li + 2H_2O \longrightarrow 2LiOH + \mathbf{H_2}$	0.18	0.34
Lithium Hydride (LiH)	$LiH + H_2O \longrightarrow LiOH + \mathbf{H_2}$	0.10	0.13
Lithium Aluminium Hydride ($LiAH_4$)	$LiAH_4 + 4H_2O \longrightarrow LiOH + Al(OH)_3 + \mathbf{4H_2}$	0.12	0.14
Sodium (Na)	$2Na + 2H_2O \longrightarrow 2NaOH + \mathbf{H_2}$	0.60	0.62
Sodium Hydride (NaH)	$NaH + H_2O \longrightarrow NaOH + \mathbf{H_2}$	0.31	0.22
Potassium (K)	$2K + 2H_2O \longrightarrow 2KOH + \mathbf{H_2}$	1.02	1.18
Calcium (Ca)	$2Ca + 2H_2O \longrightarrow 2CaOH + \mathbf{H_2}$	1.06	0.69
Calcium Carbide (CaC_2)	$CaC_2 + 2H_2O \longrightarrow Ca(OH)_2 + \mathbf{C_2H_2}$	0.22	0.10

Table 2 Gas combustion energy content

Gas	Reaction (in air 20.9% O_2 by vol.)	Combustion energy density (J/cm^3, based on HHV)
Hydrogen (H_2)	$2H_2 + O_2 \longrightarrow 2H_2O$	3.6 (100% H_2)
Acetylene (C_2H_2)	$2C_2H_2 + 5O_2 \longrightarrow 4CO_2 + 2H_{20}$	4.2 (119% H_2)
Butane (C_4H_{10})	$2C_4H_{10} + 13\ O_2 \longrightarrow 8CO_2 + 10H_2O$	3.8 (106% H_2)

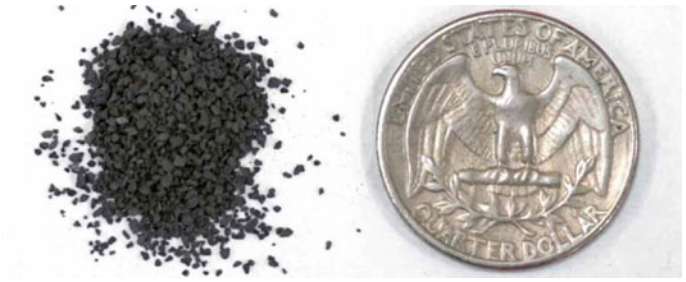


Fig. 3 Calcium Carbide: The 0.5 g of fuel shown here is sufficient for 10 launches of the 34 g thruster presented in this paper

The use of solid compounds for gas storage is common in many applications. One example is the use of sodium azide (NaN_3) decomposition to release nitrogen (N_2) for car airbag deployment, and more recently solid alkali metal hydrides have been explored by the fuel cell industry as a compact means of hydrogen storage. More relevant to aquatic propulsion are studies undertaken by the US Navy [13] examining Lithium Hydrides for use in a torpedo propulsion system, or work by [1] examining solid reactants for buoyancy control. Alkali metal hydrides have particularly high volumetric gas production per gram of reactant, and among the alkali metals and their hydrides, lithium hydride produces the largest volume of hydrogen per gram reacted with water (Table 1). However, the compound itself is acutely toxic, and an inhalation hazard, because it is generally supplied as a powder.

The alkali metals of group 1 are very malleable, and so the hazards of powder can be avoided. However, these elements have low melting points, and the reactions between group 1 elements and water is sufficiently violent to melt the metal itself and ignite the hydrogen produced, even at low quantities. This would make a fuel supply vulnerable to unplanned ignitions, and requires that the fuel be stored under oil. This would add significant complexity and weight to fuel storage and release systems. We considered lithium, sodium and potassium, the least reactive of the series. The higher weight elements have lower melting points, react too violently and some (caesium) are radioactive, and so were excluded.

Of the fuels in Table 1, calcium carbide (CaC_2, Fig. 3) and lithium aluminium hydride ($LiAH_4$) are the safest to store and easiest to handle. Of the two, $LiAH_4$ has the greatest weight efficiency, but CaC_2 has the greatest volumetric efficiency (reducing the fuel tank size and weight). In addition, the acetylene gas produced by CaC_2 has a greater combustion energy than hydrogen (Table 2). Therefore, CaC_2 was selected as a fuel source for the prototype presented in this paper.

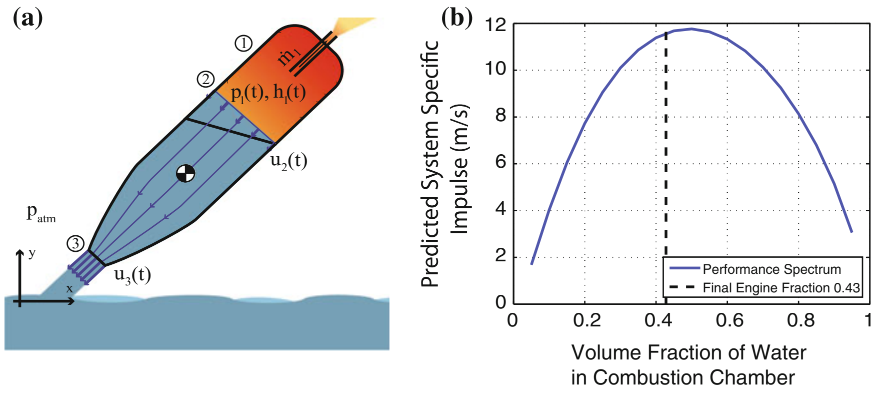


Fig. 4 **a** Jet propulsion principle: Expanding combustion products drive a jet of water through a nozzle, propelling the vehicle, with a small amount of gas lost through a choked needle. **b** The total thruster momentum change extracted from the combustion products can be maximised by finding the optimum air-water ratio in the combustion chamber using a numerical model

3 Theory

In order to size the engine and predict thrust, we created a simple model of the combustion and water expulsion. In this section we use the subscripts 1, 2 and 3 to denote variables relating to the the expanding combustion gases, the air-water interface, and the nozzle outlet respectively (Fig. 4a). The thrust force F produced by an expelled jet of mass flow $\dot{m}_3$ and velocity u_3 is given by Eq. 1.

$$F = \dot{m}_3 u_3 \tag{1}$$

The incompressibility of water means the expelled jet will be at atmospheric pressure, and the gas expansion rate will equal the water outflow (Eq. 2). The water flow with the tank is treated as quasi-1D by assuming uniform axial flow across the jet section.

$$u_3 = \dot{V}_2 / A_3 \tag{2}$$

$$A_2 u_2 = A_3 u_3 \tag{3}$$

where u is the water velocity, A_n is the jet cross sectional area and V_2 is the volume flow of water out of the tank. The unsteady form of Bernoulli's equation (Eq. 4) can be recovered from Euler's equation by integrating from the air-water interface to the nozzle exit (Fig. 4a). Total pressure along a streamline running from 2 to 3 is equal to the instantaneous gas pressure in the chamber:

$$\int_2^3 \frac{\partial u}{\partial t} ds + \frac{p_1 - p_a}{\rho_w} + \frac{1}{2}(u_3^2 - u_2^2) = 0 \tag{4}$$

where p_1 is the combustion gas pressure, p_a is atmospheric pressure and ρ_w is the density of water. To model the pressure of the combustion products driving jetting, the heat addition from combustion is assumed to take place rapidly, such that the enthalpy of the combustion gases increases by the HHV (49.92 MJ/kg fuel, [11]) instantaneously at the start of combustion. The precombustion atmosphere is mostly air (93% by volume), and so the combustion products are treated as air, obeying the ideal gas equation of state (Eq. 5). The combustion chamber has a small vent needle (inside diameter 0.6 mm) fitted to allow it to fill passively with water (Sect. 4.2), that vents a small amount of the combustion products. Flow through this needle will be choked, with mass flow given by Eq. 6. A first law energy balance is then used to account for both the work done against water pressure and the energy lost through the vent needle (Eq. 7).

$$p_1 = m_1 R T_1 / V_1 \tag{5}$$

$$\dot{m}_1 = \frac{\gamma}{\sqrt{\gamma - 1}} \left(\frac{\gamma + 1}{2} \right)^{-\frac{1}{2}\left(\frac{\gamma+1}{\gamma-1}\right)} p_1 A_{needle} / \sqrt{c_p T_1} \tag{6}$$

$$\dot{H}_1 = - p_1 \dot{V}_1 - \dot{m}_1 h_1 \tag{7}$$

where γ, c_p and R are the adiabatic index, heat capacity and gas constant of air, and H_1 and T_1 are the gas enthalpy and temperature. The inclusion of a simple conductive heat transfer model of energy loss through the chamber walls into Eq. 7 had negligible effect on predictions (1% decrease in total impulse), so the process was treated as adiabatic. The model overpredicts the total thrust production (Sect. 5), which is interpreted as being largely due to incomplete combustion and quenching of the hot combustion products by the water in the chamber.

3.1 Optimum Gas-Water Ratio

For a given combustion energy density, the volume of air before launch determines the total energy that can be recovered as thrust. Once the stored water has been expelled, further gas expansion produces negligible thrust, but if too little gas is stored, there is insufficient energy to expel all the water at significant speed. There then exists an optimum ratio of air to water. To obtain this optimum, the system specific impulse, I_{sp}, is used as an objective. This is numerically computed by integrating the predicted thrust profile with respect to time, and dividing it by the total mass of the thruster before launch, including the mass of water in the combustion chamber (Eq. 8). This maximises the momentum imparted to the thruster, and consequently the launch height that can be achieved.

$$I_{sp} = \int F dt / m_{total} \tag{8}$$

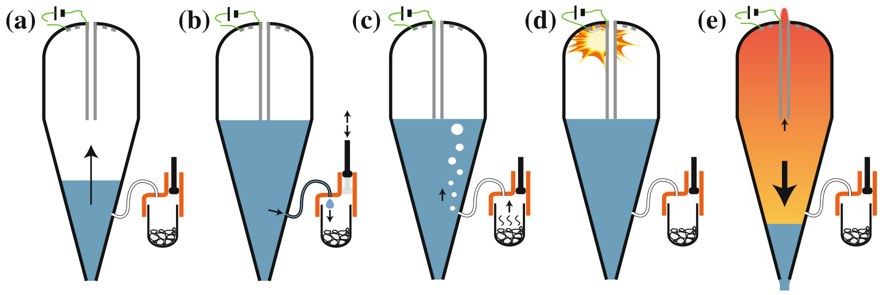


Fig. 5 Illustration of the jet firing sequence: **a** Water enters combustion chamber, with air exiting through the central needle. **b** Water reaches central needle and further water is prevented from entering. Fuel tank plunger is actuated, drawing a drop of water onto the fuel reserve. **c** Acetylene gas is produced, enters the combustion chamber and mixes with air. **d** A spark is created in the chamber, igniting the Acetylene. **e** After combustion, the high pressure products expel the water, producing thrust. Air flow through the narrow needle is choked and limited. After combustion, the sequence can be repeated, and furhter fuel reserves reacted

where m_{total} is the total mass of the vessel at the point of launch, including water. Based on the geometry of the final vessel and the HHV of acetylene, a water volume fraction of 0.45 was predicted to give the maximal performance (Fig. 4b).

4 Engine Design

In this section, we introduce the operation of the fabricated explosive water jet thruster (Fig. 5) and several key design features, including a valveless combustion chamber with passive water fill control, and an on board ignition system. The final engine design requires 4.5 ml of acetylene for stoichiometric combustion, produced by reacting 13 mg of CaC_2 with 8 µl of water. The CaC_2 used is supplied in 1 mm granules of 75% purity (Fig. 3), and the 1 ml fuel tank can hold sufficient fuel for 10 launches. Twice as much fuel as is necessary is stored in the system and further room is left in the fuel tank, because the $Ca(OH)_2$ reaction byproduct has a tendency to passivate the CaC_2 reaction after several fuellings, by preventing water from reaching unreacted CaC_2.

4.1 Fuelling System

The fuel system uses a simple syringe primer mechanism to produce Acetylene in small quantities. This system consists of a narrow (⌀ 0.5 mm) Teflon pipe connected to the combustion chamber below the water, running to the fuel tank. The fuel tank

Table 3 Thruster weight breakdown

Engine part	Mass [g]	% of total
Combustion chamber	12.1	35.5
CaC_2 Fuel (10 shots)	0.3	0.9
Fuel tank / plunger	2.5	7.3
Fuel servo	4.4	12.9
Piezo igniter	2.6	7.6
Igniter actuator / frame	3.5	10.3
Supercapacitor	4.2	12.3
Microcontroller	1.5	4.4
100 mAh LiPo battery	3.2	9.4
Total mass	**34.1**	**100**

is otherwise sealed to the outside atmosphere, and surface tension is sufficient to prevent water from entering the fuel tank through the teflon pipe (Table 3).

To force water into the fuel tank, a small plunger in the top of the tank is actuated. As the plunger is retracted, water is drawn from the combustion chamber through the pipe, onto the CaC_2 in the fuel tank. The water then reacts with the fuel, and Acetylene gas is produced. The resulting pressure pushes acetylene back through the pipe and into the combustion chamber, where it can be ignited. The 8 μl of water required is near to the smallest droplet that can be formed on the end of the teflon tube. Because of this, priming and spark delivery occur in a timed sequence, with 1 s between the introduction of a water drop into the fuel tank (the start of acetylene production) and combustion, This stops overproduction of acetylene from preventing ignition, in the event a larger droplet is aspirated.

To actuate the fuel system, the plunger is driven by a small 4.4 g servo motor, which is controlled by an Arduino microcontroller. By modifying the stroke of the plunger and the dwell time at the top of the stroke, this process can be accurately controlled so that the correct amount of water is aspirated into the fuel tank, controlling the amount of fuel reacted and consequently the volume of acetylene produced.

4.2 Combustion Chamber

The acetylene produced is then ignited in a combustion chamber partially filled with water. This chamber has been designed not only to contain the acetylene explosion, but also to control the fill level of water, ensuring that the correct gas-water ratio is maintained prior to firing (Sect. 3). This is achieved passively without the use of valves, by exploiting the choking of high pressure gas flow through a narrow needle. With the vehicle floating nose up on the water surface, air can pass through the needle, allowing water to enter the combustion chamber through the exit nozzle. The end of the needle is positioned so that it is blocked when the water reaches the desired fill point, and the chamber fills no further, leaving sufficient air volume for combustion.

The air remaining inside the chamber can then be mixed with acetylene and ignited. During engine firing, pressure inside the combustion chamber reaches up to 10 bar within 5 ms. Consequently, the outflow through the narrow needle used to allow water to enter will choke, limiting the mass flow rate through the needle. Because of this, high pressure is maintained inside the combustion chamber for a short time, and the water inside the chamber is forced through the main nozzle, generating thrust. A small amount of water is also ejected through the needle during firing, but this will only further restrict needle outflow.

The fabricated combustion chamber is constructed in two parts from high impact polystyrene. The lower section is a cylinder terminated by a 20° cone, which forms the water outflow nozzle. The upper section is a hemispherical dome which contains the ignition spark gap and the regulator needle. The regulator needle has an inside diameter of 0.6 mm. Performance would be improved if this were smaller, but narrower needles had a tendency to block with soot and prevent refilling of the chamber with water after ignitions.

4.3 Ignition System

Experiments with hotwire and spark systems showed that a spark produced ignition more reliably, and a hot wire was prone also prone to breaking during firings. The fabricated prototype uses a piezoelectric igniter, which can produce electric arcs of up to 1 cm in length. The igniter is connected to two spark gaps in series, placed at different points in the chamber, to produce two smaller sparks and ensure ignition where the air and fuel have mixed poorly. The central needle vent forms the final contact.

To produce a spark, the piezoelectric igniter requires 30 N of force over a 5 mm stroke, which is produced by two Flexinol NiTi shape memory alloy (SMA) wires in series. The wires are sheathed in teflon and the tubes sealed with grease to allow the wires to actuate underwater. Wire actuation is driven by an aerogel supercapacitor, triggered by the same microcontroller used to drive the servo. The wires are mounted through carbon fibre tubes, which support the compression forces on the igniter, and are also used to mount the other engine components (Fig. 6).

5 Combustion Tests

The operation of the fabricated final prototype was tested statically, with the combustion chamber and control components mounted to a perspex sheet, which was mounted to a fixed force sensor. During tests, the jet was mounted vertically, and immersed in a 5 l glass tank. The fuelling and ignition systems were then actuated,

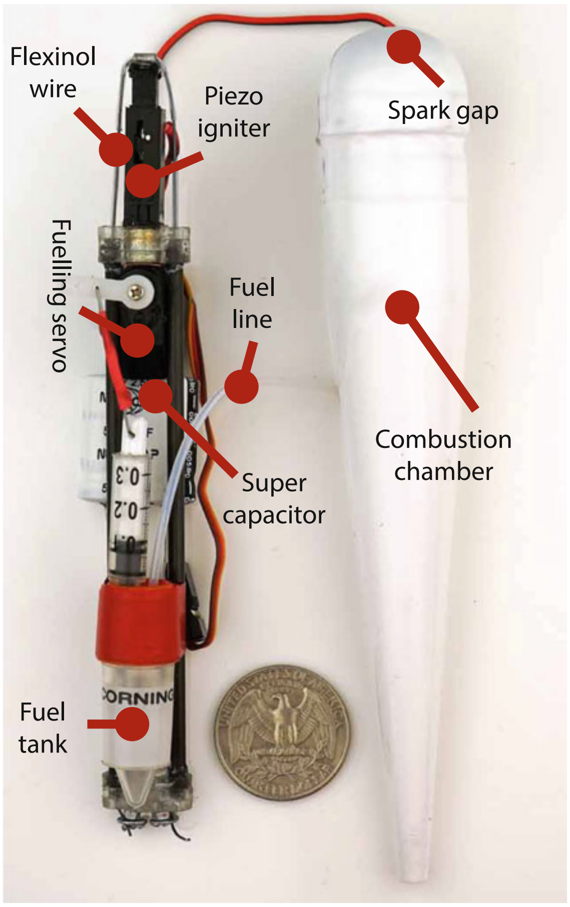


Fig. 6 Fabricated prototype, with removable fuel tank, fuel priming pump and shape memory alloy piezoelectric ignition system

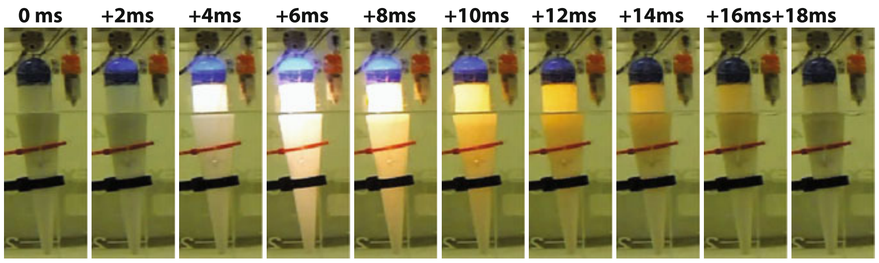


Fig. 7 Video sequence of the jet combustion, filmed at 480fps: The acetylene explosion rapidly increases the internal pressure to up to 10 bar, expelling water and producing over 20 N of thrust

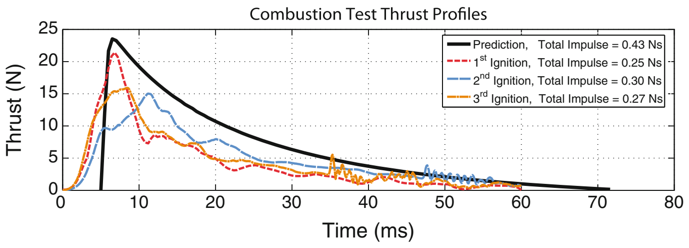


Fig. 8 Experimental Data: Static thrust measurements of prototype engine compared to theoretical thrust prediction. The prediction assumes combustion takes place instantaneously, while in practice peak pressure occurs around 5–10 ms after ignition, and the predicted thrust curve has been shifted left to offer a clearer visual comparison

force data was collected at 2500 Hz (Figs. 7, 8), and the engine was filmed at 480 fps. The recorded data shows a good level of consistency between successive actuations, although maximal pressure was not achieved repetitively. Video data showing gas bubbles leaving the nozzle exit indicated that all water was expelled during each explosion. The recorded thrust time histories were integrated to give the total impulse, which were 0.25, 0.30 and 0.27 Ns for the first, second and third ignitions respectively.

6 Discussion

6.1 Combustion Tests

The shape and duration of the measured thrust profiles compare well to theory, but total impulse is reduced by 28–40% from prediction. The fall off in thrust after the initial peak is much steeper in the measured data. This means that pressure is dropping in the chamber faster than predicted by the change in gas volume and needle outflow. This is interpreted as the both result of incomplete combustion and quenching of the hot combustion products by the water they are expelling, neither of which are accounted for by the model.

While the tests show a level of consistency sufficient for design, there is also a variation in total impulse of approximately 10% between successive tests. Examination of the data suggests that this is due to incomplete combustion, due to inhomogeneities in the pre-combustion atmosphere: In the two tests which showed less than the maximum peak pressure, oscillations in thrust at approximately 500 Hz can be observed. The oscillations are characteristic of ‘knock’ in internal combustion engines, caused by autoignition of unburned gases compressed by the combustion products. In internal combustion engines, the tendency for a fuel to autoignite is expressed in terms of octane number, with a lower number indicating a lower level of compression for autoignition, and a stronger tendency to knock. Acetylene has a Research Octane

Number (RON) of 50 [11] indicating a high susceptibility to autoignition (this can be compared to common automobile petrol with an RON of 100). The presence of secondary ignitions towards the end of the gas expansion indicates that the initial combustion was not complete, resulting in reduced pressure. This is borne out by the lack of secondary ignitions in the higher peak thrust profile.

The incomplete combustion does not have a significant effect on the overall performance, but the actuation consistency could nonetheless be improved by modification of the gas dispersal system. Discharging the acetylene through a smaller tube will build higher pressure in the fuel tank and result in smaller, more scattered gas bubbles, or the splitting of the fuel line to enter the engine at two points could increase the mix homogeneity before ignition.

6.2 Engine Design

The prototype has demonstrated the functionality of the key systems of the engine. The fuelling system is able to meter the gas accurately enough for powerful combustion, and the combustion chamber's valveless design allows repeated refuelling and ignition. The presence of water, and the short duration of the combustion means that the polystyrene combustion chamber showed no signs of melting or burning after over 20 actuations.

The most significant loss in performance comes from the choked needle, which vents energy from the combustion chamber. Making the needle narrower, to reduce this loss would improve performance, but prevents the chamber from refilling with water reliably. The losses could instead be greatly reduced by the addition of a sprung check valve with a minimum closing pressure, but this would add mass and complexity, and the functionality of a sprung valve is liable to degrade with repeated exposure to explosions, unlike the implemented choking system, which has no moving parts.

Importantly, the fuelling and ignition systems (50% of total engine mass) represent the device's 'mass capital', and only the fuel tank will increase in size if the combustion chamber is scaled up. The system could therefore be readily resized for a larger payload or higher jumps, as a mission demands and achieve increased performance to weight. The total cost of the engine is $35, including the arduino and lipo battery, enabling the device to be treated as disposable.

The fuel supply of the device is also a fraction of the total vehicle mass (0.9%), and an increase in fuel capacity would allow the engine to function as an aquatic jump-glider, with fuel for a similar number of jumps to many terrestrial jumping microrobots. This would enable aquatic locomotion over obstacles in flooded or littoral areas in a similar capacity to terrestrial jumpers. Both for gliding locomotion and propelled flight, the engine design lends itself well to integration onto an airframe. The ignition and fuel systems are both only linked to the chamber by flexible wires and tubes, and so may be freely positioned within a vehicle, facilitating both stable mass distribution and compact design. This is also important because the filling and ignition sequence described in Sect. 4.2 relies on the vehicle being upright, and the ability to position mass to achieve this passively will be important.

6.3 Comparison to Another Water Jet Thruster

Siddall and Kovač [17] previously demonstrated aquatic escape with a similar system, using a tank of compressed CO_2 instead of an acetylene explosion to drive the expulsion of water. This system weighed 40 g, and was able to propel a 100 g flying robot to a speed of 12 m/s from beneath the water. Comparing the thrust profiles from [17] with the measured thrust from the acetylene engine (Fig. 9) it can be seen that while the peak thrust is much higher, the acetylene engine has a smaller total impulse. However, the CO_2 thruster can only produce a single shot, and when the 10 total shots of the combustion prototype are taken into account, the mass efficiency is much greater (Table 4). With further design improvements, and an increase in engine size, the combustion prototype is expected to be a far better system for aquatic escape.

6.4 Other Potential Combustible Fuel Sources

Fuels were chosen principally based on their source reactant, rather than the performance of their product gas. Other combustible gases such as methane and hydrogen

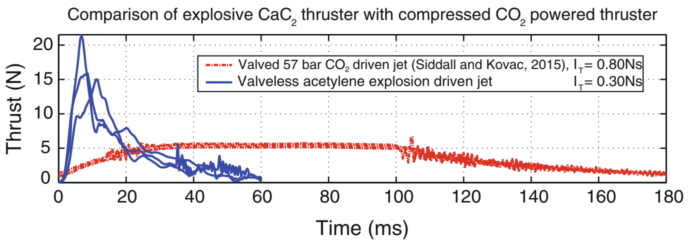


Fig. 9 Comparison of the explosive thruster presented here with the single shot CO_2 aquatic escape thruster presented by [18]

Table 4 Explosive CaC_2 water jet system compared to the compressed CO_2 thruster presented by [18]

Thruster type	Mass (g)	Peak thrust (N)	Total impulse (per shot) (Ns)	System specific impulse (inc. total shots) (ms^{-1})
Compressed CO_2 thruster	40.1	5.1	0.80	19
Explosive CaC_2 thruster	34.1	21.5	0.30	88

have a much better resistance to autoignition and may ultimately be better fuel choices for the design if consistency is a strong design consideration. In addition, hydrogen and methane can be sourced directly from the environment, rather than a fuel reservoir, methane through the use of methanogenic bacteria, and hydrogen by direct electrolysis. The latter process also generates pure O_2 and so could better use the available combustion volume. Either of these approaches would potentially created a fully renewable energy source for a miniature robot, while the engine presented here has a limited fuel capacity, although it is comparatively very compact.

While hydrogen and methane as fuels offer the potential for fully renewability, acetylene has an different advantage in that it is also one of the most detonateable fuels. If a narrow tube of fuel and air is ignited at one end, the flame front will accelerate down the tube, and if the front is able to travel a sufficient distance, it will transition to a travelling normal shockwave (deflagration to detonation transition). This shockwave precompresses the reactants and results in much higher combustion pressures, on the order of 20 bar, which has the potential to enhance the maximum power output of the device. By modifying the chamber geometry, the calculations of [19] applied to combustion on a similar scale to the prototype presented here suggest that a deflagration to detonation transition of the flame front would be achievable in a high aspect ratio combustion chamber, if the fuel could be appropriately premixed. This will be explored further in the future.

7 Conclusions

In this paper we have demonstrated the use of CaC_2 as a fuel source to repeatably generate explosions in a miniature robot. These explosions can be harnessed to produce very high power to weight ratios, which are otherwise unachievable with conventional miniature systems. The fabricated thruster is entirely self contained and weighs only 34 g, with sufficient fuel for 10 controlled explosions. The device scales well, and an the power or fuel capacity of the thruster could be increased without significantly increasing the total system mass.

The fuelling and ignition processes used for the thruster could also be used to increase the performance of other systems, such as a pneumatic piston or soft robot. The production of gas from a solid reactant offers a highly compact means of gas storage, even more so when environmental water is used as a reactant. The solid liquid reaction used is also an accurate means of metering gas production, as it is easier to accurately dispense liquids than gases. If the produced is not ignited, a similar solid reactant system could also be used in inflatable structure deployment.

Acknowledgements This project was funded by the UK Engineering and Physical Sciences Research Council and an Imperial College London Faculty of Engineering Undergraduate Research Opportunities Programme (UROP) Award.

References

1. Borchsenius, J., Pinder, S.: Underwater glider propulsion using chemical hydrides. In: OCEANS 2010 IEEE-Sydney, IEEE, pp. 1–8 (2010)
2. Churaman, W., Currano, L.J., Morris, C.J., Rajkowski, J.E., Bergbreiter, S., et al.: The first launch of an autonomous thrust-driven microrobot using nanoporous energetic silicon. J. Microelectromech. Syst. **21**(1), 198–205 (2012)
3. Desbiens, A.L., Pope, M.T., Christensen, D.L., Hawkes, E.W., Cutkosky, M.R.: Design principles for efficient, repeated jumpgliding. Bioinspir. Biomim. **9**(2), 025009 (2014)
4. Ijspeert, A.J., Crespi, A., Ryczko, D., Cabelguen, J.-M.: From swimming to walking with a salamander robot driven by a spinal cord model. Science **315**(5817), 1416–1420 (2007)
5. Izraelevitz, J., Triantafyllou, M.: A novel degree of freedom in flapping wings shows promise for a dual aerial/aquatic vehicle propulsor. arXiv preprint arXiv:1412.3843 (2014)
6. Jones, K., Boria, F., Bachmann, R., Vaidyanathan, R., Ifju, P., Quinn, R.: Mmalv - the morphing micro air-land vehicle. In: IROS 2006 (2006)
7. Liang, J., Yang, X., Wang, T., Yao, G., Zhao, W.: Design and experiment of a bionic gannet for plunge-diving. J. Bionic Eng. **10**(3), 282–291 (2013)
8. Lock, R.J., Burgess, S.C., Vaidyanathan, R.: Multi-modal locomotion: from animal to application. Bioinspir. Biomim. **9**(1), 011001 (2014)
9. Lock, R.J., Vaidyanathan, R., Burgess, S.C.: Impact of marine locomotion constraints on a bio-inspired aerial-aquatic wing: experimental performance verification. J. Mech. Robot. **6**(1), 011001 (2014)
10. Loepfe, M., Schumacher, C.M., Lustenberger, U.B., Stark, W.J.: An untethered, jumping roly-poly soft robot driven by combustion. Soft Robot. **2**(1), 33–41 (2015)
11. McAllister, S., Chen, J., Fernandez-Pello, A.: Fundamentals of Combustion Processes. Mechanical Engineering Series. Springer, Berlin (2011)
12. Meadows, G., Atkins, E., Washabaugh, P., Meadows, L., Bernal, L., Gilchrist, B., Smith, D., Van Sumeren, H., Macy, D., Eubank, R., et al.: The flying fish persistent ocean surveillance platform. In AIAA Unmanned Unlimited Conference (2009)
13. Newhouse, H., Payne, P.: Underwater power source study. Technical report, DTIC Document (1981)
14. Ore, J.-P., Elbaum, S., Burgin, A., Zhao, B., Detweiler, C.: Autonomous aerial water sampling. In: The 9th International Conference on Field and Service Robots (FSR) (2013)
15. Schwarzbach, M., Laiacker, M., Mulero-Pazmany, M., Kondak, K.: Remote water sampling using flying robots. In: 2014 International Conference on Unmanned Aircraft Systems (ICUAS), IEEE, pp. 72–76 (2014)
16. Siddall, R., Kovač, M.: Launching the aquamav: bioinspired design for aerial-aquatic robotic platforms. Bioinspir. Biomim. **9**(3), 031001 (2014)
17. Siddall, R., Kovač, M.: Fast aquatic escape with a jet thruster. IEEE/ASME Trans. Mechatron. (2016)
18. Siddall, R., Kovač, M.: A water jet thruster for an aquatic micro air vehicle. In: 2015 IEEE International Conference on Robotics and Automation (ICRA), IEEE (2015)
19. Silvestrini, M., Genova, B., Parisi, G., Leon Trujillo, F.J.: Flame acceleration and ddt run-up distance for smooth and obstacles filled tubes. J. Loss Prev. Process Ind. **21**(5), 555–562 (2008)
20. Tolley, M., Shepherd, R.F., Karpelson, M., Bartlett, N.W., Galloway, K.C., Wehner, M., Nunes, R., Whitesides, G.M., Wood, R.J.: An untethered jumping soft robot. In: 2014 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2014), IEEE, pp. 561–566 (2014)
21. Vidyasagar, A., Zufferey, J.-C., Floreano, D., Kovač, M.: Performance analysis of jump-gliding locomotion for miniature robotics. Bioinspir. Biomim. (2015)
22. Weiss, P.: Hop hop hopbots!: designers of small, mobile robots take cues from grasshoppers and frogs. Sci. News **159**(6), 88–91 (2001)
23. Woodward, M.A., Sitti, M.: Multimo-bat: a biologically inspired integrated jumping gliding robot. Int. J. Robot. Res. (2014)

A Global Strategy for Tailsitter Hover Control

Robin Ritz and Raffaello D'Andrea

1 Introduction

There is an increasing demand to apply hover-capable flying machines to long-range and long-endurance missions. Traditional multicopters use rotors to produce lift and overcome gravity, which is, compared to fixed-wing airplanes, inefficient in terms of both required energy per distance flown and required energy per flight time, and thus a limiting factor of the operating range and flight duration [1]. In order to overcome this limitation, powered lift aircrafts [2] such as the so-called *tailsitter* vehicle [3, 4] have been suggested. A tailsitter is able to take off and land vertically on its tail with the nose and thrust direction pointing upwards. For fast forward flight, the vehicle tilts to a near-horizontal attitude resulting in a more efficient lift production with conventional wings. Compared to other powered lift aircraft types (such as tiltrotors [5, 6] or tiltwings [7, 8]), the major advantage of a tailsitter is its mechanical simplicity; no mechanism for changing the direction of the propulsive system has to be added, saving weight and reducing susceptibility to malfunctions. As a result of the availability of cheap, lightweight electronic components and the numerous potential applications of such small hybrid vehicles,[1] many researchers and companies have recently started research programs exploring the capabilities of

[1]We refer *hybrid vehicles* to vehicles that provide both hover-capabilities and wings for aerodynamic lift production.

This research was supported by the Hans-Eggenberger Stiftung and the Swiss National Science Foundation.

R. Ritz (✉) · R. D'Andrea
Institute for Dynamic Systems and Control, ETH Zürich,
Sonneggstrasse 3, 8092 Zürich, CH, Switzerland
e-mail: rritz@ethz.ch

R. D'Andrea
e-mail: rdandrea@ethz.ch

A. Bicchi and W. Burgard (eds.), *Robotics Research*, Springer Proceedings
in Advanced Robotics 2, DOI 10.1007/978-3-319-51532-8_2

these flying machines. For example, the company Transition Robotics is selling the tailsitter vehicle Quadshot for the hobby and research market [9]. In August 2014, the team of Google's Project Wing tested a tailsitter prototype for a packet delivery service vehicle.[2] However, in March 2015 Google announced that the tailsitter wing design approach was scrapped; the project leaders came to the conclusion that it is still too difficult to control such a vehicle in a reliable and robust manner.[3]

Over the last decades, the research community has developed many successful control strategies for small unmanned aerial vehicles including quadcopters and conventional fixed-wing airplanes (see for example [10, 11] and references therein). However, relatively little attention has been paid to small powered lift aircrafts such as tailsitters, where the large flight envelope and the highly nonlinear dynamics introduce additional challenges for control design. The problem of attitude control for tailsitters or similar vehicles has been addressed, for example, in [12–17]. In order to avoid singularities, typically a quaternion representation for the vehicle's attitude is used, combined with linear feedback on the quaternion error vector to obtain the desired body rates. However, alternative approaches of representing and controlling attitude exist, such as the resolved tilt-twist method leading to better tracking performance when large attitude errors occur [15, 16]. As computational units become more powerful and less expensive, model predictive control (MPC) [18] has become a viable approach for controlling systems with fast dynamics such as small flying machines; first results of an MPC-based controller for a tailsitter in hover flight have been published in [19].

In this paper, we address the problem of designing a nonlinear hover controller for a small flying wing tailsitter vehicle, which should be capable of recovering to hover from any initial attitude. The challenges of this task lie in the fact that typically the vehicle's actuators operate close to their saturation limits, and in the fact that the rotation axis along the wing is weakly actuated and might be dominated by aerodynamic torques. As mentioned above, most traditional tailsitter attitude control methods are based on linear quaternion feedback or similar strategies, which provide under some assumptions global asymptotic stability. However, for large attitude errors these methods have difficulties to properly account for the different magnitudes of attainable torques around the different rotation axes. Due to the nonlinear nature of the attitude dynamics, these algebraic feedback laws typically do not result in an optimal maneuver. In order to overcome this limitation, we propose a control strategy which plans trajectories in such a way that the vehicle exploits its strongly actuated axis (the axis actuated by the propellers) if it has to recover from large attitude errors. The control law is obtained by solving an optimal control problem that minimizes a quadratic cost with a particular structure. Since the computation is not feasible in real-time, solutions for a set of initial attitudes are precomputed and stored in a map. For each controller update, the optimal inputs are then read from this lookup table, and fed to the system in an MPC-like manner. We do not prove

[2]http://www.bbc.com/news/technology-28964260/.

[3]http://blogs.wsj.com/digits/2015/03/17/google-working-on-new-drone-after-wing-design-failed/.

stability of the presented controller, however the performance of the control strategy is analyzed in a simulation environment and the results indicate that the tailsitter is able recover to hover from any initial attitude, given that the initial velocity does not exceed a certain limit. Furthermore, the effectiveness of the control strategy is demonstrated in the ETH Zurich Flying Machine Arena.

The remainder of the paper is structured as follows: In Sect. 2 we present a dynamic model for a small flying wing tailsitter. Section 3 introduces a nonlinear hover control strategy for the tailsitter vehicle. Simulations and experimental results are presented in Sect. 4, and we conclude in Sect. 5.

2 Flying Wing Tailsitter Model

In this section, we derive a model for the dynamics of a small flying wing tailsitter vehicle. Due to the large operating range and complex aerodynamic forces and torques acting on the vehicle, deriving an adequate dynamic model for a tailsitter is a challenging task. Typically, the complex aerodynamic properties are obtained either by CFD methods [17, 20], by measurement series covering all relevant operating points [21], or by first-principles models combined with heuristics that capture some of the unmodeled effects [22]. Since herein we focus on control design, we follow a similar approach as proposed in [22] and derive a first-principles model of the considered tailsitter vehicle.

The vehicle is actuated by two propellers, one in front of each wing, and two flaps located at the wings' trailing edges. An illustration of the tailsitter is shown in Fig. 1. The control inputs are the propeller forces $f_{prop,l}$ and $f_{prop,r}$, and the flap angles $\delta_{flap,l}$ and $\delta_{flap,r}$. All four control inputs are subject to saturations:

$$\begin{aligned} f_{prop,min} &\leq f_{prop,l}, f_{prop,r} &&\leq f_{prop,max}, \\ \delta_{flap,min} &\leq \delta_{flap,l}, \delta_{flap,r} &&\leq \delta_{flap,max}. \end{aligned} \tag{1}$$

We assume that the vehicle's airspeed is small, such that the range of attainable propeller forces $[f_{prop,min}, f_{prop,max}]$ can be considered to be constant. The propellers are of fixed-pitch type and the motors are not able to reverse direction mid-flight, meaning that the propellers cannot produce negative thrust, i.e. $f_{prop,min} > 0$.

We introduce a body-fixed coordinate frame B with origin at the vehicle's center of mass, as shown in Fig. 1. The z-axis of the body frame B is aligned with the thrust direction, the x-axis points along the left wing, and the y-axis completes the right-handed coordinate system. We denote unit vectors along the axes of the body frame as $\vec{e}_x^{\,B}$, $\vec{e}_y^{\,B}$, and $\vec{e}_z^{\,B}$, respectively. The position of the vehicle's center of mass relative to an inertial coordinate frame I, expressed in this inertial frame I, is denoted as ${}_I\vec{p} = (p_x, p_y, p_z)$. (In order to simplify notation, vectors may be expressed as n-tuples $\vec{x} = (x_1, x_2, \ldots, x_n)$, with dimension and stacking clear from context.) The tailsitter's attitude is described by a unit quaternion $\vec{q} = (q_0, q_1, q_2, q_3)$ that represents a rotation from the inertial frame I to the body-fixed frame B. For more

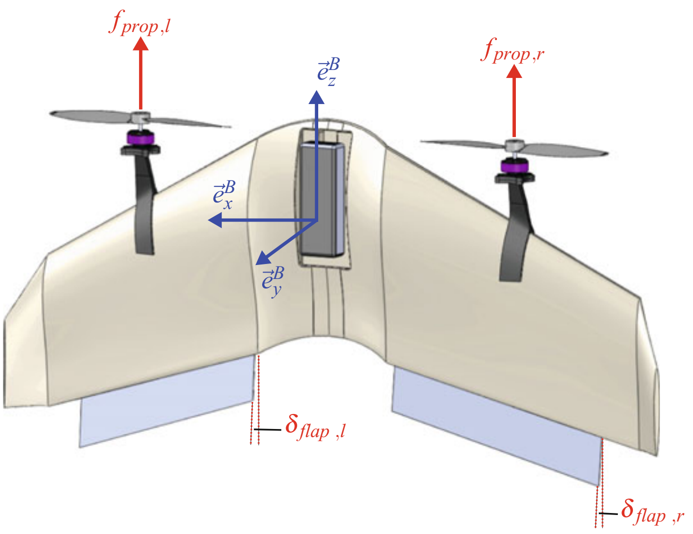


Fig. 1 Illustration of a flying wing tailsitter vehicle. The vehicle is actuated by two propellers that produce forces along the body z-axis, and two flaps that produce aerodynamic forces by deflecting the airflow over the wings

information on representing attitudes with unit quaternions, see for example [23] and references therein. For the remainder of the paper, unless otherwise stated, we will express all quantities in the body frame B. If a vector $\vec{x}$ is expressed in the inertial frame I, the notation ${}_I\vec{x}$ will be used. The translational velocity of the body frame B relative to the inertial frame I is denoted as $\vec{v} = (v_x, v_y, v_z)$, and the rotational body rates are denoted as $\vec{\omega} = (\omega_x, \omega_y, \omega_z)$. The position and attitude kinematics are

$$\begin{aligned} {}_I\dot{\vec{p}} &= \boldsymbol{R}^T \vec{v}, \\ \dot{\vec{q}} &= \frac{1}{2} \boldsymbol{W}^T \vec{\omega}, \end{aligned} \tag{2}$$

where $\boldsymbol{R}$ denotes the rotation matrix from the inertial frame I to the body frame B, and $\boldsymbol{W}$ is the quaternion rate matrix [23]. The vehicle is modeled as a rigid body with mass M and rotational inertia $\boldsymbol{J}$, and the dynamics are thus given by the Newton–Euler equations:

$$\begin{aligned} M\dot{\vec{v}} &= \vec{f}_{tot} - \vec{\omega} \times M\vec{v}, \\ \boldsymbol{J}\dot{\vec{\omega}} &= \vec{m}_{tot} - \vec{\omega} \times \boldsymbol{J}\vec{\omega}, \end{aligned} \tag{3}$$

where $\vec{f}_{tot}$ and $\vec{m}_{tot}$ denote the external force and torque vector, respectively, acting on the vehicle. The external force is modeled as

$$\vec{f}_{tot} = \vec{f}_{air} + (f_{prop,l} + f_{prop,r})\vec{e}_z^B - Mg\vec{e}_z^I, \tag{4}$$

where $\vec{f}_{air}$ denotes the aerodynamic force, g is the gravitational acceleration, and $\vec{e}_z^B$ and $\vec{e}_z^I$ are the unit vectors along the z-axis of the corresponding frame.

Similarly, the external torque acting on the vehicle is modeled as

$$\vec{m}_{tot} = \vec{m}_{air} + x_{prop}(f_{prop,r} - f_{prop,l})\vec{e}_y^B + \kappa_{prop}(f_{prop,r} - f_{prop,l})\vec{e}_z^B, \tag{5}$$

where $\vec{m}_{air}$ denotes the aerodynamic torque, x_{prop} is the x-axis offset of the propellers, and κ_{prop} is the propellers' torque-to-thrust ratio (the left propeller rotates counter-clockwise, and the right propeller rotates clockwise). Note that we assume that the offset of the propellers in the y-direction is negligible.

Typically, the aerodynamic force $\vec{f}_{air}$ and torque $\vec{m}_{air}$ are complex functions of the vehicle state and control input. However, in order to keep the problem tractable and applicable to the MPC-like control strategy that will be introduced in Sect. 3, we apply the following first-principles aerodynamic model: For each wing the aerodynamic force and torque is computed separately, assuming that the air velocity is uniform over the wing, and assuming that there is no cross coupling between the left and right wing. First, we compute the total velocity of the corresponding wing:

$$\vec{v}_{wing} = \vec{v} - \vec{p}_{wing} \times \vec{\omega}, \tag{6}$$

where $\vec{p}_{wing}$ denotes the reference position of the wing relative to the body frame B. The propellers of the tailsitter are mounted in front of the wings; consequently the air is accelerated along the negative body z-axis if the propellers produce a positive thrust force. For simplicity, we assume that the entire wing is in the propeller streamtube. For low airspeeds, the z-component of the total airspeed including propeller-induced speedup can be approximated using Momentum Theory [24]:

$$v_{wing,tot,z} = \sqrt{\frac{2f_{prop}}{\rho_{air}A_{prop}} + v_{wing,z}^2}, \tag{7}$$

where ρ_{air} denotes the density of air, and A_{prop} is the propeller area. We neglect that the Momentum Theory approximation (7) becomes less accurate as $v_{wing,z}$ becomes negative. The angle of attack α_{wing} and the reference airspeed $v_{wing,ref}$ of the corresponding wing are then defined as [25]

$$\alpha_{wing} = -\text{atan2}(v_{wing,y}, v_{wing,tot,z}), \quad v_{wing,ref} = \sqrt{v_{wing,y}^2 + v_{wing,tot,z}^2}. \tag{8}$$

The aerodynamic force is modeled as

$$\vec{f}_{wing} = \vec{c}_{air}(\alpha_{wing}, \delta_{flap})v_{wing,ref}^2, \tag{9}$$

where $\vec{c}_{air}$ denotes a coefficient vector, and the arguments $(\alpha_{wing}, \delta_{flap})$ are stated explicitly in order to highlight that the coefficients are a function of angle of attack and flap angle. When modeling the aerodynamic coefficients $\vec{c}_{air}$, we make two simplifying assumptions; (1) the wing does not produce a force component along the body x-axis, and (2) the flap deviates the airflow behind the wing by a small angle proportional to the flap angle deviation δ_{flap}:

$$\vec{c}_{air}(\alpha_{wing}, \delta_{flap}) = \left(c_y(\alpha_{wing}) + c_{y,\delta}\delta_{flap}\right)\vec{e}^{\,B}_y + c_z(\alpha_{wing})\vec{e}^{\,B}_z, \tag{10}$$

where the function $c_y(\alpha_{wing})$ corresponds to the lift coefficient of the wing, the constant $c_{y,\delta}$ describes first order effects of flap angle deviations on the lift coefficient, and the function $c_z(\alpha_{wing})$ corresponds to the drag coefficient of the wing. Similarly, the aerodynamic torque is modeled as

$$\vec{m}_{wing} = \vec{d}_{air}(\alpha_{wing}, \delta_{flap})v^2_{wing,ref}, \tag{11}$$

with

$$\begin{aligned}\vec{d}_{air}(\alpha_{wing}, \delta_{flap}) = &\left(d_x(\alpha_{wing}) + d_{x,\delta}\delta_{flap}\right)\vec{e}^{\,B}_x + d_y(\alpha_{wing})\vec{e}^{\,B}_y \\ &+ \left(d_z(\alpha_{wing}) + d_{z,\delta}\delta_{flap}\right)\vec{e}^{\,B}_z,\end{aligned} \tag{12}$$

where the functions $d_x(\alpha_{wing})$, $d_y(\alpha_{wing})$, and $d_z(\alpha_{wing})$ are wing characteristics, and the constants $d_{x,\delta}$ and $d_{z,\delta}$ describe first order effects of flap angle deviations. This completes the derivation of the first-principles model for the aerodynamic effects acting on one of the tailsitter's wings. The total aerodynamic forces and torques yield

$$\begin{aligned}\vec{f}_{air} &= \vec{f}_{wing,left} + \vec{f}_{wing,right}, \\ \vec{m}_{air} &= \vec{m}_{wing,left} + \vec{m}_{wing,right}.\end{aligned} \tag{13}$$

Note that due to symmetry considerations the aerodynamic coefficients $c_{(\cdot)}$ and $d_{(\cdot)}$ are identical for both wings, except for some sign changes where appropriate.

3 Control Strategy

In this section, we present a control strategy for the tailsitter modeled in Sect. 2. First, a desired attitude and a desired thrust force is computed, and subsequently an attitude controller computes desired body rates in order to track the desired attitude. The desired body rates are then fed to an inner control loop that computes actuator commands.

3.1 Desired Attitude and Thrust Force

As proposed in [26], a desired acceleration ${}_I\vec{a}_{des}$ is computed based on position error $\Delta\vec{p}$ and velocity error $\Delta\vec{v}$. (The errors are expressed in the inertial frame I.)

The control loop is shaped such that for each coordinate the system behaves like a second-order system with some desired time constant and damping ratio. In order to compensate for modeling errors and external disturbances such as wind, an additional integral state $\dot{\vec{s}} = \Delta\vec{p}$ is added. Thus, the desired acceleration is given by

$$_I\vec{a}_{des} = \boldsymbol{K}_s\vec{s} + \boldsymbol{K}_p\Delta\vec{p} + \boldsymbol{K}_v\Delta\vec{v}, \tag{14}$$

where the control gains $\boldsymbol{K}_s$, $\boldsymbol{K}_p$, and $\boldsymbol{K}_v$ are computed such that the desired closed-loop properties are met. Using the substitution $_I\vec{f}_{tot} = M_I\vec{a}_{des}$, the desired thrust vector can then be computed by rearranging (4):

$$_I\vec{f}_{thrust,des} = (f_{prop,l} + f_{prop,r})_I\vec{e}_z^B = M_I\vec{a}_{des} + Mg_I\vec{e}_z^I - _I\vec{f}_{air}. \tag{15}$$

Note that, in order to compute the desired thrust vector, the controller needs an estimate of the current aerodynamic force $_I\vec{f}_{air}$, which could be an estimate or simply the current nominal value. Since the thrust force acts along $\vec{e}_z^B$, we choose the desired attitude such that the body z-axis is aligned with the desired thrust direction. A desired attitude that aligns actual and desired thrust axis is given by

$$\vec{q}_{thrust,des} = \left(\cos(\theta_{des}/2), \vec{n}_{des}\sin(\theta_{des}/2)\right), \tag{16}$$

where θ_{des} is the desired tilt angle, and $\vec{n}_{des}$ the desired tilt rotation direction:

$$\theta_{des} = \arccos\left({}_I\vec{e}_z^I \cdot {}_I\vec{e}_{z,des}^B\right), \quad \vec{n}_{des} = \frac{{}_I\vec{e}_z^I \times_I \vec{e}_{z,des}^B}{||{}_I\vec{e}_z^I \times_I \vec{e}_{z,des}^B||}, \tag{17}$$

where the desired body z-axis is given by

$${}_I\vec{e}_{z,des}^B = \frac{{}_I\vec{f}_{thrust,des}}{||{}_I\vec{f}_{thrust,des}||}. \tag{18}$$

After this tilt alignment, we can rotate the vehicle around its z-axis without changing the thrust direction, hence we can choose an arbitrary yaw angle ψ_{des}. The desired attitude yields

$$\vec{q}_{des} = \vec{q}_{yaw,des} \cdot \vec{q}_{thrust,des}, \tag{19}$$

where $(\cdot)$ denotes the quaternion multiplication and $\vec{q}_{yaw,des}$ is given by

$$\vec{q}_{des,yaw} = \left(\cos(\psi_{des}/2), 0, 0, \sin(\psi_{des}/2)\right). \tag{20}$$

The desired thrust force is given by the magnitude of the desired thrust vector:

$$f_{thrust,des} = ||{}_I\vec{f}_{thrust,des}||. \tag{21}$$

3.2 Attitude Control

As mentioned above, one of the challenges when designing a tailsitter attitude controller is to cope with the limited torques that can be produced around the body x-axis, since the corresponding lever arm is small and the flap saturation boundaries are relatively tight. In addition, large undesired aerodynamic torques may act on the x-axis, which further complicates the controller design. On the other hand, the attainable torques around the y-axis are relatively high, since this torque is produced by the propellers' differential thrust. In the following, we propose a method for computing desired body rates in order to control the vehicle's attitude, while turning preferably around the better actuated y-axis.

The attitude error is given by

$$\vec{q}_{err} = \vec{q}_{est} \cdot (\vec{q}_{des})^{-1}, \tag{22}$$

where $\vec{q}_{est}$ denotes the current estimated attitude of the vehicle. For convenient notation, we will drop the error subscript in the following, i.e. we define $\vec{q} = \vec{q}_{err}$. We assume that an inner control loop perfectly tracks the body rates $\vec{\omega}$, such that we can directly set the body rates without any delay or dynamics. As mentioned in Sect. 2, the error quaternion kinematics are given by

$$\dot{\vec{q}} = \frac{1}{2}\boldsymbol{W}^T\vec{\omega}. \tag{23}$$

The objective of the proposed attitude controller is to align the vehicle's thrust axis (i.e. the body z-axis) with the desires thrust direction. (The remaining degree of freedom, i.e. the yaw angle of the vehicle, is controlled separately and not considered here.) We define the tilt angle θ to be the angle between the desired and actual thrust direction; it is given by

$$\theta = \arccos\left(q_0^2 - q_1^2 - q_2^2 + q_3^2\right). \tag{24}$$

In order to control the tilt angle θ to zero, we choose the following cost function to be minimized:

$$J = \int_{t_0}^{t_f} c_\theta\theta^2 + (c_x + c_{x,\theta}\theta^2)\omega_x^2 + c_y\omega_y^2 + c_z\omega_z^2 \,\mathrm{dt}, \tag{25}$$

where $c_{(\cdot)}$ indicates constant, positive weight parameters, and the different terms are explained in the following: The first term $c_\theta\theta^2$ quadratically penalizes the tilt angle that should be controlled to zero. The second term $(c_x + c_{x,\theta}\theta^2)\omega_x^2$ penalizes the control effort around the x-axis. As mentioned above, the attainable torques around this axis are subject to tight bounds; therefore we want to avoid that large tilt errors are corrected by turning around this axis. Thus, the weight on the control input ω_x is not constant, but contains a term that grows quadratically with the tilt angle θ, such that large errors are corrected by turning mainly around the other two axes. Finally, the third and fourth term $c_y\omega_y^2$ and $c_z\omega_z^2$, respectively, penalize the inputs around the remaining two axes.

For a given initial attitude error $\vec{q}_{ini}$, the optimal control inputs $\vec{\omega}^*$ solve the optimization problem

$$\begin{aligned} \text{minimize} \quad & J \\ \text{subject to} \quad & \dot{\vec{q}} = \frac{1}{2}\boldsymbol{W}^T\vec{\omega}, \\ & \vec{q}(t_0) = \vec{q}_{ini}, \\ & \vec{\omega}(t) \in \mathbb{R}^3 \ \forall\, t \in [t_0, t_f]. \end{aligned} \tag{26}$$

In order to simplify this optimization problem, we leverage Pontryagin's Minimum Principle [27] to derive necessary conditions for optimality, which are then used for computing candidate optimal solutions. The Hamiltonian of the above problem is given by

$$H = c_\theta\theta^2 + (c_x + c_{x,\theta}\theta^2)\omega_x^2 + c_y\omega_y^2 + c_z\omega_z^2 + \frac{1}{2}\vec{\lambda}^T\boldsymbol{W}^T\vec{\omega}, \tag{27}$$

where $\vec{\lambda} = (\lambda_0, \lambda_1, \lambda_2, \lambda_3)$ denotes the costate vector. The costate equation $\dot{\vec{\lambda}} = -\nabla_{\vec{q}}H$ yields

$$\begin{aligned} \dot{\lambda}_0 &= -\tfrac{\mathrm{d}}{\mathrm{d}q_0}H = (\lambda_0 k - \lambda_1\omega_x - \lambda_2\omega_y - \lambda_3\omega_z)/2, \\ \dot{\lambda}_1 &= -\tfrac{\mathrm{d}}{\mathrm{d}q_1}H = (-\lambda_1 k + \lambda_0\omega_x - \lambda_3\omega_y + \lambda_2\omega_z)/2, \\ \dot{\lambda}_2 &= -\tfrac{\mathrm{d}}{\mathrm{d}q_2}H = (-\lambda_2 k + \lambda_3\omega_x + \lambda_0\omega_y - \lambda_1\omega_z)/2, \\ \dot{\lambda}_3 &= -\tfrac{\mathrm{d}}{\mathrm{d}q_3}H = (\lambda_3 k - \lambda_2\omega_x + \lambda_1\omega_y + \lambda_0\omega_z)/2, \end{aligned} \tag{28}$$

with the shorthand notation

$$k = \frac{8(c_\theta + c_{x,\theta}\omega_x^2)\theta}{\sqrt{1 - (q_0^2 - q_1^2 - q_2^2 + q_3^2)^2}}. \tag{29}$$

Since the final state $\vec{q}(t_f)$ is free and costless, the costates satisfy the final condition $\vec{\lambda}(t_f) = (0, 0, 0, 0)$. According to the Minimum Principle, the optimal inputs $\vec{\omega}^*$ minimize the Hamilton over the set of attainable controls. We do not impose any constraints on the body rates $\vec{\omega}$, and the cost function is quadratic in $\vec{\omega}$ with positive weights $c_{(\cdot)}$. Thus, an expression for the optimal body rates $\vec{\omega}^*$ can be obtained by setting the gradient of the Hamiltonian with respect to $\vec{\omega}$ to zero:

$$\begin{aligned}\omega_x^* &= \frac{-\lambda_1 q_0 + \lambda_0 q_1 + \lambda_3 q_2 - \lambda_2 q_3}{4(c_x + c_{x,\theta}\theta^2)},\\ \omega_y^* &= \frac{-\lambda_2 q_0 - \lambda_3 q_1 + \lambda_0 q_2 + \lambda_1 q_3}{4c_y},\\ \omega_z^* &= \frac{-\lambda_3 q_0 + \lambda_2 q_1 - \lambda_1 q_2 + \lambda_0 q_3}{4c_z}.\end{aligned} \tag{30}$$

By substituting the expression for the optimal body rates (30) into the quaternion kinematics (23), the optimization problem (26) can be written as a boundary value problem (BVP):

$$\begin{aligned}\dot{\vec{q}} = \frac{1}{2}\boldsymbol{W}^T\vec{\omega}^*, \quad \dot{\vec{\lambda}} = -\nabla_{\vec{q}} H,\\ \vec{q}(t_0) = \vec{q}_{ini}, \quad \vec{\lambda}(t_f) = \vec{0}.\end{aligned} \tag{31}$$

Thus, we can obtain candidate optimal solutions to the optimization problem (26) by numerically solving BVP (31).[4]

On an Intel i7-3520M processor, it typically takes between 10 and 30 s to compute a solution to BVP (31), which is not fast enough for real-time applications. Therefore, we compute trajectories for a set of initial attitudes $\{\vec{q}_{ini}\}$ and create a lookup table for the candidate optimal body rates $\vec{\omega}^*$ at the beginning of the trajectory. Since in the cost function (25) the yaw angle is not penalized, we can arbitrarily rotate the inertial reference frame I around its z-axis without changing the body rates $\vec{\omega}^*$ of the corresponding solution to BVP (31). Consequently, we can always rotate the reference frame I such that the quaternion error around the z-axis is zero, i.e. $q_{ini,3} = 0$. Hence, the state space for which control inputs need to be computed is two-dimensional. Each point in this reduced two-dimensional state space is defined by a tilt angle $\theta \in [0, \pi]$ and a tilt direction $\phi \in [0, 2\pi]$, where a tilt rotation around the x-axis corresponds to $\phi = 0$, and a tilt rotation around the y-axis corresponds to $\phi = \pi/2$. Symmetry considerations indicate that it is sufficient to compute body rates for $\phi \in [0, \pi/2]$, and map these solutions onto the full range $\phi \in [0, 2\pi]$ using appropriate coordinate transformations. We thus choose a uniformly sampled set over the space $\{(\theta, \phi) \mid \theta \in [0, \pi], \phi \in [0, \pi/2]\}$, which defines the set of initial attitudes $\{\vec{q}_{ini}\}$ for which solutions to BVP (31) are computed.

Figure 2 shows the resulting candidate optimal body rates $\vec{\omega}^*$ as function of tilt angle θ and tilt direction ϕ, for a map that was computed as described above. For each controller update, the desired body rates $\vec{\omega}_{des}$ for the current attitude error are read from the precomputed maps using linear interpolation and subsequently sent to the inner control loop. As we can see in Fig. 2, the maps for the desired body rates $\vec{\omega}_{des}$ are relatively smooth. Hence, we could approximate the desired body rates by fitting some particular functions with a small number of parameters into the maps. For example in an onboard implementation where memory is limited, this approach might be beneficial.

[4]Computations are executed with Matlab [28], using the function *'bvp4c'*.

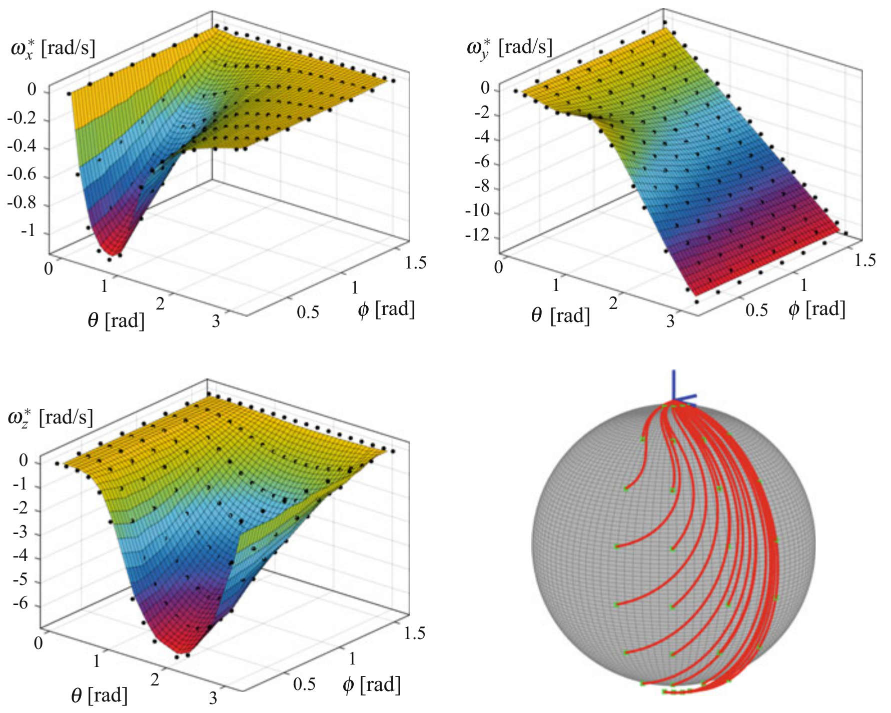


Fig. 2 Map of candidate optimal body rates $\vec{\omega}^*$ for the x-axis (*top left*), y-axis (*top right*), and z-axis (*bottom left*). The *black dots* indicate elements of $\{\vec{q}_{ini}\}$ for which BVP (31) has been solved. The *bottom right* drawing shows a selection of candidate optimal tilt trajectories. The *top* of the sphere corresponds to $\theta = 0$, and the *bottom* to $\theta = \pi$. We can observe that the trajectories do not correspond to the shortest rotation towards zero tilt, because the body rates around the different axes have different weights in the cost function (25)

3.3 Body Rate Control

An inner body rate controller tracks the desired body rates $\vec{\omega}_{des}$ using rate gyroscope measurements $\vec{\omega}_{meas}$. First, the range of allowed propeller forces is adjusted to the current flight situation, and then the four actuator commands are computed.

3.3.1 Propeller Force Boundaries

As mentioned in Sect. 2, due to actuator saturations the attainable propeller forces are constrained to the range $f_{prop} \in [f_{prop,min}, f_{prop,max}]$. However, in order to ensure that the flaps do not loose effectiveness, we pose two additional constraints on the propeller forces: Firstly, since the torques produced by the flaps scale quadratically with the reference airspeed $v_{wing,ref}$ (as defined in (8)), we choose a minimum required reference airspeed over the wings. For each wing, this defines an additional lower bound on the propeller force f_{prop}, which can be computed using (7) and (8) and depends on the current velocity and body rates of the vehicle. Secondly, we choose a maximum allowable angle of attack α_{wing} (as defined in (8)). Doing so, we can avoid that the flaps loose their effectiveness due to stall phenomena at large angles of attack. Again, this defines an additional lower bound on the propeller force f_{prop} given by (7) and (8) and depending on the current state of the vehicle. Even though these two additional constraints might narrow the range of allowed propeller forces considerably, experimental results show that, to some extent, the benefits of effective flaps outweigh this drawback.

3.3.2 Actuator Commands

The body rate controller is designed such that the elements of the body rate error $(\vec{\omega}_{meas} - \vec{\omega}_{des})$ follow three decoupled first order systems with desired time constants $\vec{\tau}_\omega = (\tau_{\omega,x}, \tau_{\omega,y}, \tau_{\omega,z})$. The total desired torque acting on the vehicle is obtained by rearranging the angular dynamics (3), which yields

$$\vec{m}_{des} = \boldsymbol{J}(\vec{\omega}_{des} - \vec{\omega}_{meas})/\vec{\tau}_\omega + \vec{\omega}_{meas} \times \boldsymbol{J}\vec{\omega}_{meas}, \tag{32}$$

where the division is executed element-wise. By inspection of (5) and (11), we find that the total torque around the body y-axis can be written as

$$m_{tot,y} = x_{prop}(f_{prop,r} - f_{prop,l}) + d_y(\alpha_{wing,l})v^2_{wing,ref,l} - d_y(\alpha_{wing,r})v^2_{wing,ref,r}, \tag{33}$$

and does not depend on the flap angles $\delta_{flap,l}$ and $\delta_{flap,r}$. Note that both α_{wing} and $v_{wing,ref}$ depend on the corresponding propeller force f_{prop}. The two propeller forces $f_{prop,r}$ and $f_{prop,l}$ can thus be computed such that the two conditions

$$m_{tot,y} = m_{des,y}, \quad f_{prop,l} + f_{prop,r} = f_{thrust,des}, \tag{34}$$

are satisfied. However, due to actuator saturations, we might not be able to satisfy both conditions of (34). In this case, we adapt the desired thrust $f_{thrust,des}$ such that we can achieve the desired torque $m_{des,y}$; intuition and experiments indicate that it is more important to align the thrust axis, rather than keep the desired thrust value. The flap angles $\delta_{flap,l}$ and $\delta_{flap,r}$ are then computed such that the remaining two desired torques are achieved, i.e. such that

$$m_{tot,x} = m_{des,x}, \quad m_{tot,z} = m_{des,z}. \tag{35}$$

Again, if we cannot satisfy both conditions, we adapt $m_{des,z}$ such that we can achieve $m_{des,x}$ which is crucial for aligning the thrust axis and thus considered to be more important.

Note that, as we can see from (9), the aerodynamic force $\vec{f}_{air}$ depends on the actuator inputs, hence the actual aerodynamic force may differ from the nominal or estimated value we have used in (15) to compute the desired thrust vector $\vec{f}_{thrust,des}$. However, we assume that the difference is negligible.

4 Results

In this section, we present simulation and experimental results of the proposed tailsitter hover control strategy.

4.1 Simulation Results

During controller design we made some significant assumptions, in order to keep the problem tractable. We assume, for example, that the inner control loop perfectly tracks the desired body rates even though the actuators saturate quickly, which might have a significant effect on the behavior of the vehicle. Therefore, we verify the reliability performance of the proposed controller by simulating its behavior for a variety of initial conditions. In particular, we simulate the vehicle's dynamics for a set of 10'000 random initial conditions with the following probability distribution: The magnitude of the initial velocity vector $\vec{v}_{ini}$ is uniformly distributed between 0 and 5 m/s, and its direction is uniformly distributed among all possible directions. Similarly, the magnitude of the initial body rates $\vec{\omega}_{ini}$ is uniformly distributed between 0 and 10 rad/s, and its direction is uniformly distributed among all directions. Further, the initial attitude $\vec{q}_{ini}$ is uniformly distributed over the whole attitude space, and the initial position $\vec{p}_{ini}$ is chosen to be the origin. For each of the 10'000 simulations, we draw a random sample for the initial state, and subsequently simulate the vehicle's dynamics forward in time. The task of the vehicle is to recover and to fly back to the origin. We consider the maneuver to be successful, if this task is achieved within reasonable thresholds. We found that for all 10'000 simulations the recovery is successful, indicating that for the chosen range of initial states, the proposed control strategy is able to cope with the neglected effects. However simulations also show that for larger initial velocities, due to the dominant aerodynamic forces and torques, the vehicle might enter a tumbling motion pattern and is not always able to stabilize. Thus, for these regions of the state space, the proposed hover controller is not suitable.

4.2 Experimental Results

The experiments are carried out in the ETH Zurich Flying Machine Arena [26]. The vehicle consists of a Robbe Mini Wing RC styrofoam airframe, a PX4 FMU flight computer [29], two MKS DS65K servos actuating the flaps, two Hacker A05–13S brushless electric motors driving the propellers, two ZTW Spider 12A ESCs with SimonK firmware [30], communication radios, and a LiPo battery. The testbed provides an infrared motion-capture system, which is used for estimating the tailsitter's current state. The outer control loop that computes desired body rates, as presented in Sect. 3, is implemented offboard and runs on a desktop workstation. The desired body rates are sent over wireless radios to the vehicle with a rate of 50 Hz. Hence, the inner body rate controller is implemented onboard, and it runs at a rate of 1000 Hz.

In order to show the performance of the control strategy, we execute recovery maneuvers from arbitrary initial states back to hover flight. Therefore, the vehicle is manually thrown into the air, and at a predefined height of 2.5 m the controller is switched on. Figure 3 shows the body rate and tilt trajectories of such a recovery maneuver. Further, a video showing a series of throws and subsequent recoveries back to hover can be found on http://www.youtube.com/watch?v=JModZfnVAv4. The experiments indicate that for most of these manual throws the vehicle is able to recover; the only observed reason for failure is the limited space of the Flying Machine Arena, i.e. the vehicle bounces into the floor during the recovery maneuver.

Because of the relatively large drag coefficient perpendicular to the wings, a hovering tailsitter is susceptible to horizontal wind gusts. In order to analyze the performance of the proposed control strategy for such a situation, we simulate an

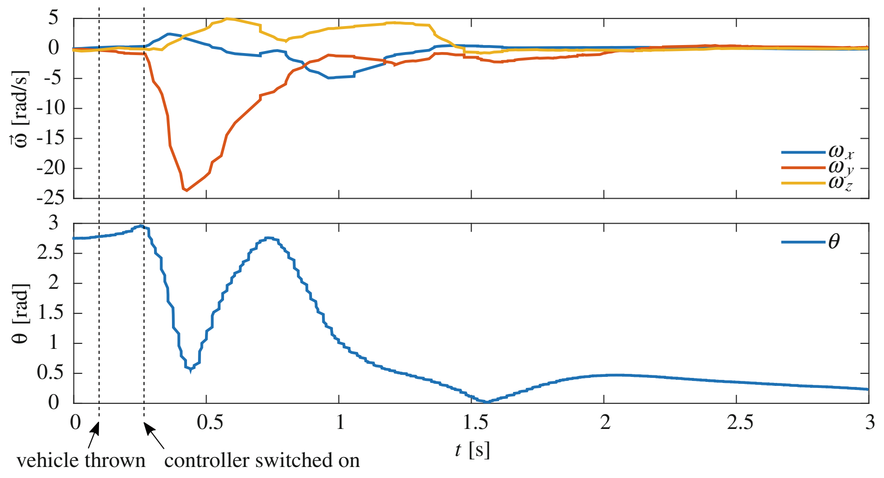


Fig. 3 Body rate and tilt angle trajectories of an example recovery maneuver. The vehicle is thrown into the air and the controller is switched on at the indicated points in time. Roughly 1 s after the controller has been switched on, the vehicle has recovered to near-hover flight

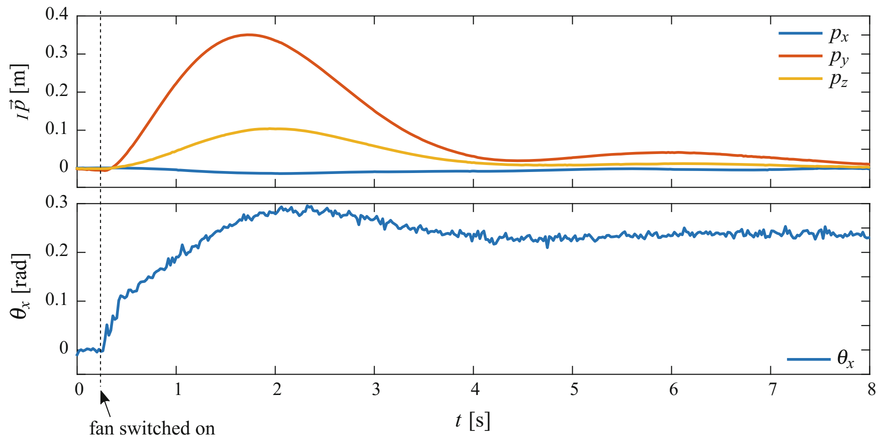


Fig. 4 Position and pitch angle trajectories when a wind gust hits the vehicle. Initially, the vehicle is hovering in resting air and the body frame B is aligned with the inertial frame I. At the indicated point in time a fan is switched on, which results in an airstream of about 2 m/s along the y-axis. The plot on the *top* shows that due to the integrator term of the position controller, the vehicle compensates the external disturbance without a steady-state position error. In the *bottom* plot, θ_x denotes the pitch angle around the body x-axis, and we can observe that the vehicle pitches in order to counteract the external disturbance

external disturbance using a fan that creates an airstream with a velocity of about 2 m/s. The direction of the airstream is aligned with the body y-axis, i.e. perpendicular to the wings. Figure 4 shows the response of the vehicle to such a simulated wind gust.

5 Conclusion

An approach for a tailsitter hover controller has been presented, and its performance has been analyzed in simulation and verified in experiments. The experimental results indicate that the proposed tailsitter hover control strategy performs well under external aerodynamic disturbances, and is able to stabilize the vehicle from any initial attitude, given that the initial velocity is below a certain limit.

Future work includes the extension of the control method such that it can be applied to high-velocity maneuvers, for example by adding a high-level trajectory planner. Further, we would like to investigate if a proof of stability can be established.

References

1. Filippone, A.: Flight Performance of Fixed and Rotary Wing Aircraft. Elsevier, Oxford (2006)
2. Deckert, W.H., Franklin, J.A.: Powered-lift aircraft technology. Technical report, NASA (1989)
3. Campbell, J.P.: Research on VTOL and STOL aircraft in the United States. In: Proceedings of the First International Congress in the Aeronautical Sciences, Advances in Aeronautical Sciences, Madrid, 8–13 September, 1958, vol. 2. Pergamon Press (1959)
4. Woods, R.J.: Convertiplanes and Other VTOL Aircraft. Technical report, SAE Technical paper (1957)
5. Wernicke, K.G.: Tilt Prop-rotor Composite Research Aircraft. Technical report, DTIC Document (1968)
6. Lichten, R.L.: Some Aspects of Convertible Aircraft Design. J. Aeronaut. Sci. (Inst. Aeronaut. Sci.), **16**(10) (2012)
7. Stuart, J.: TiltWing Propelloplane Potentialities. J. Am. Helicopter Soc., **4**(1) (1959)
8. Tosti, L.P.: Flight Investigation of the Stability and Control Characteristics of a 1/4-Scale Model of a Tilt-Wing Vertical-Take-Off-and-Landing Aircraft. Technical report, NASA (1959)
9. Sinha, P., Esden-Tempski, P., Forrette, C.A., Gibboney, J.K., Horn, G.M.: Versatile, modular, extensible VTOL aerial platform with autonomous flight mode transitions. In: IEEE Aerospace Conference. IEEE (2012)
10. Powers, C., Mellinger, D., Kumar, V.: Quadrotor kinematics and dynamics. In: Handbook of Unmanned Aerial Vehicles. Springer, Heidelberg (2014)
11. Sujit, P., Saripalli, S., Sousa, J.B.: Unmanned aerial vehicle path following: a survey and analysis of algorithms for fixed-wing unmanned aerial vehicles. IEEE Control Syst., **34**(1) (2014)
12. Knoebel, N.B., McLain, T.W.: Adaptive quaternion control of a miniature tailsitter UAV. In: American Control Conference (ACC). IEEE (2008)
13. Johnson, E N., Wu, A., Neidhoefer, J.C., Kannan, S.K., Turbe, M.A.: Flight-test results of autonomous airplane transitions between steady-level and hovering flight. J. Guidance Control Dyn., **31**(2) (2008)
14. Kita, K., Konno, A., Uchiyama, M.: Transition between level flight and hovering of a tail-sitter vertical takeoff and landing aerial robot. Adv. Robot., **24**(5–6) (2010)
15. Matsumoto, T., Kita, K., Suzuki, R., Oosedo, A., Go, Hoshino, K.Y., Konno, A., Uchiyama, M.: A hovering control strategy for a tail-sitter VTOL UAV that increases stability against large disturbance. In: IEEE International Conference on Robotics and Automation (ICRA). IEEE (2010)
16. Beach, J.M., Argyle, M.E., McLain, T.W., Beard, R.W., Morris, S.: Tailsitter attitude control using resolved tilt-twist. In: International Conference on Unmanned Aircraft Systems (ICUAS). IEEE (2014)
17. Jung, Y., Cho, S., Shim, D.H.: A comprehensive flight control design and experiment of a Tail-Sitter UAV. In: AIAA Guidance, Navigation, and Control Conference (GNC) (2013)
18. Camacho, E.F., Alba, C.B.: Model predictive control. Springer Science & Business Media, Heidelberg (2013)
19. Anderson, P., Stone, H.: Predictive guidance and control for a tail-sitting unmanned aerial vehicle. In: Information, Decision and Control (IDC). IEEE (2007)
20. Stone, R.H.: Aerodynamic modeling of the wing-propeller interaction for a tail-sitter unmanned air vehicle. J. Aircr., **45**(1) (2008)
21. Erickson, G.E.: High angle-of-attack aerodynamics. Annu. Rev. Fluid Mech., **27**(1) (1995)
22. Knoebel, N.B., Osborne, S.R., Snyder, D.O., McLain, T.W., Beard, R.W., Eldredge, A.M.: Preliminary modeling, control, and trajectory design for miniature autonomous tailsitters. In: AIAA Guidance, Navigation, and Control Conference (GNC) (2006)
23. J. Diebel, Representing attitude: Euler angles, unit quaternions, and rotation vectors, Stanford University, Tech. Rep., 2006
24. W. Johnson, Helicopter theory. Courier Corporation, 2012

25. P. H. Zipfel, Modeling and Simulation of Aerospace Vehicle Dynamics (AIAA Education). American Institute of Aeronautics & Astronautics, 2003
26. S. Lupashin, M. Hehn, M. W. Mueller, A. P. Schoellig, M. Sherback, and R. D'Andrea, A platform for aerial robotics research and demonstration: The Flying Machine Arena, Mechatronics, **24**(1), pp. 41–54, 2014
27. H. P. Geering, Optimal Control with Engineering Applications. Springer, 2007
28. The MathWorks Inc., Matlab R2012a (7.14.0.739), 2012
29. PX4 FMU. http://www.pixhawk.ethz.ch/px4/modules/px4fmu (2017). Accessed 27 Jan 2017
30. SimonK - Open Source Firmware for ATmega-based Brushless ESCs. https://github.com/sim-/tgy(2015) accessed 27 April 2015

Autonomous Flights Through Image-Defined Paths

Tien Do, Luis C. Carrillo-Arce and Stergios I. Roumeliotis

1 Introduction and Related Work

In order for a quadrotor to autonomously navigate within a *known*, GPS-denied area (e.g., indoors), it must be able to find where it is, determine the path towards its goal, and control itself to follow the desired trajectory. One way to solve this problem would be to construct, typically offline, a metric 3D map of the environment that the quadrotor will then use to (i) identify where it is (i.e., localize), and (ii) compute a path towards its destination. Creating *dense* 3D maps that can be used for path planning, however, is quite challenging, especially for large buildings, due to the computational cost of the mapping process.

An alternative approach to this problem, is to represent a building using visual data collected beforehand, and describe a path as *a sequence of images* that the quadrotor needs to follow in order to reach its destination. The main advantages of an image-space representation of a path within a building are *scalability* (no metric global map needs to be constructed) and *ease of implementation* (the images can be collected by a person walking throughout the building with, e.g., a cell phone). On the other hand, though, controlling a quadrotor to follow a visual path becomes significantly more challenging due to the lack of scale and geometric information in the reference trajectory.

This work was supported by the AFOSR (FA 9550-10-1-0567).

T. Do (✉) · L.C. Carrillo-Arce · S.I. Roumeliotis
University of Minnesota, Minneapolis, MN 55455, USA
e-mail: doxxx104@umn.edu

L.C. Carrillo-Arce
e-mail: carrillo@umn.edu

S.I. Roumeliotis
e-mail: stergios@cs.umn.edu

A. Bicchi and W. Burgard (eds.), *Robotics Research*, Springer Proceedings in Advanced Robotics 2, DOI 10.1007/978-3-319-51532-8_3

Controlling a robot to reach a specific destination defined in the image space can be achieved using visual servoing (VS) [7, 8]. Most VS approaches can be classified into two categories: (i) Position-based VS (PBVS), where the control input is computed directly using a relative position, up to scale, and orientation (pose) estimate; and (ii) Image-based VS (IBVS), where the control input is determined in the image domain, while often it is assumed that the depth to the scene is, at least approximately, constant [7]. Prior work on VS for quadrotors equipped with a downward-pointing camera has addressed the problem of landing on a known target [6, 23] and hovering over an arbitrary target [3]. Furthermore, for quadrotors equipped with a forward-pointing camera, [5] classifies the environment into corridors, stairs, or “other” in order to determine the appropriate turn, side-ways, or upward motion so that the robot can continue exploration.

In the context of navigating along a visual path, VS techniques have been employed for *ground* robots (e.g., [9, 12, 13, 24]), while some of these techniques ([9, 12]) have been applied to aerial vehicles [11, 25]. In particular, in [25] an extension of the “funnel”-lane concept of [9] to 3D is presented and applied to controlling a quadrotor. Specifically, the geometric constraints based on the image coordinates of the reference features are used for determining the funnel region within which the robot should be moving in order to match the reference image. Then, the desired motion of the quadrotor is computed as the convex combination of the heading/height required for staying within the funnel region and the one the quadrotor had followed during the training phase. As criterion for switching to the next reference image, an error measure is defined based on the root mean square of the difference in the feature’s pixel coordinates between the reference and the current image. In [11], the VS method of [12] is extended to the case of a quadrotor following a visual path comprising a sequence of keyframe images selected, during the experiment, by a person. In contrast to 3-view-geometry-based approaches (e.g., [13, 17]), [11] uses the PBVS algorithm described in [8] for controlling the quadrotor. This method does not require triangulating points but instead, given sufficient baseline, it uses epipolar geometry for estimating the relative pose between the current and the reference camera frames.

A key limitation of both [11, 25] is that they cannot deal with rotations in place (often required for navigating through tight spaces), or, for the case of [11], with translations through areas with only faraway features (e.g., featureless corridors). Moreover, in both cases, the quadrotor followed rather short and fairly simple (in terms of the motions required) paths comprising a short translation and a wide turn in [25], or no turn in [11], where the quadrotor was moving back and forth between two locations connected via a direct path.

In this work, our objective is to enable a quadrotor to autonomously navigate inside a large-scale building by following a pre-recorded sequence of images that correspond to long ($\sim$75 m) and challenging (in terms of the motions involved and the type of scenes encountered) paths. To do so, our PBVS method *concurrently* minimizes the relative heading and baseline between the current and the reference images. In particular, we match features between these two images and use the 2pt[1] and the 5pt RANSACs' estimates to determine the state of the system and control the quadrotor.

The key advantages of the proposed PBVS algorithm are as follows: (i) We employ a geometry-based algorithm which computes and compares the 2pt versus the 5pt RANSAC inliers for selecting the next reference image. In contrast, [11, 25] rely on the features' pixel coordinates, which are unreliable when dealing with features at various depths; (ii) Our algorithm is capable of dealing with a wide range of conditions, such as motion along lengthy corridors, open spaces, and rotations in place, and in environments where the appearance-based feature matching may return unreliable results; (iii) Our approach does not require recording the images by manually controlling the robot through the reference paths as is done in [11, 25]. Instead, one can easily define desired paths by simply walking through the area of interest carrying a cell phone or the quadrotor. Lastly and, in order to demonstrate the efficiency, accuracy, and robustness of the proposed algorithm, we have implemented it on two quadrotors: The Parrot Bebop [4] and the DJI F450, the latter carrying a cell phone. During testing, the quadrotors operated inside challenging environments (specular reflections, large lighting surfaces), comprising long paths, tight turns, and in the presence of dynamic obstacles.

2 Quadrotors and Objective

Both quadrotors have attitude-stabilization controllers, which take as input information from an observer that processes gyroscope and accelerometer measurements, from the onboard inertial measurement unit (IMU), to estimate the roll and pitch angles, yaw-rate, and thrust of the quadrotor. Additionally, each quadrotor carries a downward-pointing camera to estimate optical flow, and an ultrasonic sensor to measure the distance to the ground. Note that despite the availability of metric information from the velocity estimated based on the optical flow and the distance to the scene, we do not use it to triangulate features and create a local map as it can be both unreliable[2] and computationally expensive. Furthermore, both quadrotors have access to (i) forward pointing wide field of view (WFOV) cameras (mounted in the front of the Bebop, or as part of the cell phone carried by the DJI) for collecting images and (ii) processors for executing in real time all image-processing and control

[1]The 2pt RANSAC estimates the relative orientation ${}^{I_1}_{I_2}\mathbf{R}$ between two images, I_1 and I_2, under the assumption of very small baseline compared to the depth of the scene. A closed-form solution for the 2pt minimal case is provided in Sect. 6.2, while the analytical solution for the least-squares solver is presented in [21]. The 5pt RANSAC [26] estimates the relative orientation ${}^{I_1}_{I_2}\mathbf{R}$ and the unit vector of translation ${}^{I_1}\mathbf{t}_{I_2}$ between two images I_1 and I_2.

[2]Under low-light conditions, the velocity measurements are reliable only for a fixed tilt angle of the vehicle. Note that when in motion, the quadrotor changes its roll and pitch which causes image blurriness (due to the increased exposure) and, hence, large errors in the optical-flow estimates.

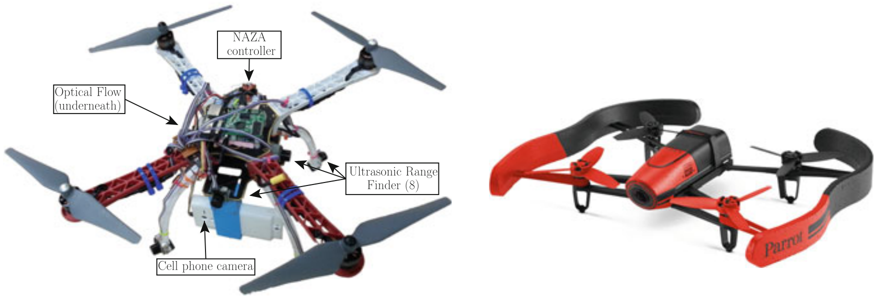


Fig. 1 The DJI F450 quadrotor (*left*) equipped with a NAZA controller, an optical-flow sensor, ultrasonic sensors, and a cell phone. The Parrot Bebop quadrotor (*right*) equipped with a 180° WFOV camera, an optical-flow sensor, and an ARM-based processor

algorithms necessary by the proposed PBVS method. Finally, and to increase safety, the DJI quadrotor carries 8 ultrasonic sensors spaced 45° apart and aligned with its arms and legs (see Fig. 1).

As mentioned earlier, the objective of this work is to develop a robust algorithm[3] that will allow the quadrotors to follow long and complex visual paths, defined as sequences of pre-recorded images between the start and final desired locations.

3 Technical Approach

Our approach comprises two phases. In the first (offline) phase, a visual-graph-based representation of the area of interest is constructed using images collected by a person walking through it. Then, given a start and an end pair of images, a feasible visual path is *automatically* extracted from the graph along with motion information (path segments that include significant translational motion or only rotations in place). In the second (online) phase, our PBVS algorithm controls the quadrotor to successively minimize the relative rotation and baseline between the images captured by its onboard camera and the corresponding reference images of the visual path. Additionally, and in order to increase robustness, our navigation approach employs a vocabulary tree (VT) [27] for relocalizing inside the previously-constructed visual graph when losing track of the reference image path. Lastly, we include an obstacle avoidance routine for the DJI so as to increase safety.

[3]Note that although both the embedded controller and the cell phone contain IMUs, which can be used, in conjunction with the camera, to form a vision-aided inertial navigation system [18], in this work, we intentionally focus on a "light", in terms of processing, vision-only approach so as to assess its performance and use it as a baseline for future comparisons.

3.1 Offline Phase

3.1.1 Map Generation

A person carrying a cell phone, or a quadrotor, walks through the area of interest collecting images at 30 Hz. Subsequently, we extract FREAK image points [2] and employ a VT to generate the visual map which is represented as a visual graph (VG) whose nodes correspond to the recorded images. An edge between two images signifies that these were matched by the VT and at least 30 point-correspondences passed the 5 or 2pt RANSAC. Furthermore, we assign *weights* to these edges inversely proportional to the number of common features (inlier matches) found between linked images. This choice is justified by the fact that the VG will be used to determine paths that the quadrotor can reliably navigate through in the image space towards its destination. This process is depicted in Fig. 2.

The VG is constructed in a matter of minutes even for large areas containing tens of thousands of images. Moreover, it can be easily updated by adding/replacing subsets of images corresponding to new/altered regions of a building.

3.1.2 Path Specification

The VG is used for computing paths between the quadrotor's start and end locations, possibly via intermediate points. Specifically, consider the graph shown in

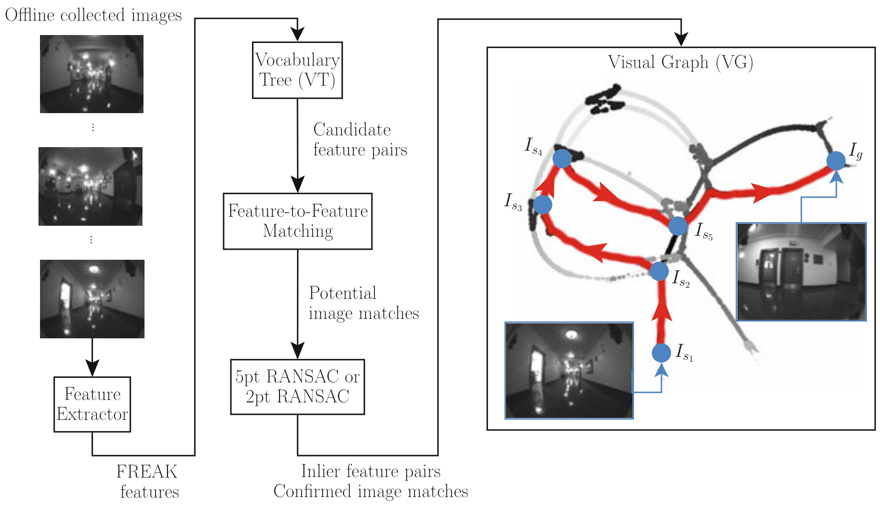


Fig. 2 Offline phase: The area of interest is described as a visual graph (VG) whose nodes correspond to images, while edges link images containing a sufficient number of common features for reliably visually-servoing between them. In the VG, I_{s1}, I_g denote the start and goal images, respectively, while $I_{s2}, \ldots, I_{s5}$ signify intermediate goal locations along the quadrotor's path specified by the user

Fig. 2. Assume that the quadrotor knows its current location (e.g., it is provided by the user, automatically determined using the VT, or saved from the previous run) corresponding to image node I_s. Then, the user specifies a destination image I_g in the VG and the reference path is determined automatically by employing Dijkstra's algorithm [10]. This process is easily extended to include intermediate locations (e.g., $I_{g_1}, I_{g_2} \dots I_{g_n}$), by simply resetting as the start of the next path segment the end image of the previous one (e.g., $I_{s_{i+1}} = I_{g_i},\ i = 1 \dots n$).

Once the path is extracted from the VG, we prune images that are very close to each other and only keep the ones that have substantial translational and/or rotational motion between them. To do so, we use an iterative process that starts from the reference image $I_1^r = I_s$ and moves along the path matching FREAK features using both the 5pt and 2pt RANSAC algorithms until it finds the first image, $I_{s+m}, m \geq 1$, that either has more 5pt than 2pt inliers, or the relative yaw angle between them is greater than 10°. In the first case, we declare that the quadrotor is in translation, otherwise, in rotation and set I_{s+m}, as the next reference image I_2^r. The resulting path $\mathscr{P} = \{I_1^r, I_2^r, \dots, I_n^r\}$ is provided to the quadrotor along with two additional pieces of information: (i) We specify which images correspond to rotation-only motion and provide the yaw angle between consecutive rotation-only images; (ii) We provide the FREAK features extracted from each reference image along with their coordinates. The former is useful in case the quadrotor gets lost (see Sect. 3.2.4), while the latter is used by the quadrotor for efficiently finding and matching its next reference image through the process described hereafter.

3.2 *Online Phase*

3.2.1 System State Determination

When there is sufficient baseline (as in [11]), and in order to minimize the relative motion between I_t (the image taken by the quadrotor's onboard camera at time-step t) and a reference image $I_k^r \in \mathscr{P}$, we use the 5pt RANSAC to estimate the 5 dof, ${}^{I_k^r}_{I_t}\hat{\mathbf{R}}$, ${}^{I_k^r}\hat{\mathbf{t}}_{I_t}$, desired motion. This estimate, however, is not reliable when the baseline between I_t and I_k^r is short [8]. Furthermore, the appearance-based feature matching between I_t and I_k^r (the 5pt RANSAC's input), is not always reliable (e.g., due to adverse lighting conditions and/or occlusions).

To address these challenges, we model our system as a hybrid automaton $\mathscr{H}$ as follows:

Definition 1: $\mathscr{H} = (\mathscr{L}, \mathbf{x}, \mathscr{E})$, where:

- $\mathscr{L}$ is the set of discrete states including:
  - ℓ_0: wide baseline (nominal condition)
  - ℓ_1: short baseline (rotation in place is necessary or reference-image switching)
  - ℓ_2: lost mode due to, e.g., motion overshoot or failure in the appearance-based feature matching.

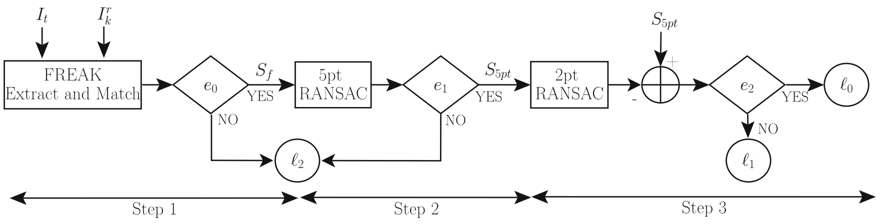


Fig. 3 Online: Schematic diagram of the steps and transitions between the different states of the automaton $\mathscr{H}$

- $\mathbf{x}(t, k)$ is the abstract state vector with elements $\mathbf{x}(t, k) = [I_t, I_k^r, \mathbf{r}(t, k)]$ where $\mathbf{r}(t, k)$ is the desired motion for minimizing the relative pose between I_t and I_k^r.
- $\mathscr{E}$ is the set of relations governing transitions between the states in $\mathscr{L} = \{\ell_0, \ell_1, \ell_2\}$.

Given $\mathscr{H}$, and in order to complete $\mathscr{P}$, the system must ideally iterate between two steps until the last element of $\mathscr{P}$ is reached: (i) In case of state ℓ_0, we compute the motion $\mathbf{r}$ and control the quadrotor so as to bring the system to state ℓ_1 (see Sect. 3.2.2); (ii) When at ℓ_1, and if there is no significant rotation between I_t and I_k^r, we switch I_k^r to the next reference image in $\mathscr{P}$ (see Sect. 3.2.3), and the system returns to state ℓ_0. In case of external disturbances, the system may reach state ℓ_2. In this case, a recovery procedure will be executed to attempt to bring the system back to ℓ_0 or ℓ_1 (see Sect. 3.2.4).

In order to accurately classify the state of the system as ℓ_0, ℓ_1, or ℓ_2 based on I_t and I_k^r, we use the process summarized in Fig. 3, and define the relations in $\mathscr{E} = \{e_0, e_1, e_2\}$ in the following 3 steps.

Step 1: We first extract and match (Hamming distance between binary descriptors less than 60) FREAK features in I_t and I_k^r, and define as $S_f(I_t, I_k^r)$ the set of all feature correspondences. Note that if the condition for sufficient feature matches $e_0 : |S_f| \geq 25$, where $|S_f|$ is the cardinality of the set S_f, is satisfied, then the system proceeds to Step 2 of the current state, else it transitions to state ℓ_2 (see Fig. 3).

Step 2: Given the bearing vectors, ${}^{I_t}\mathbf{b}_f$ and ${}^{I_k^r}\mathbf{b}_f$, from both camera frames, I_t and I_k^r, to each feature f, we employ the 5pt RANSAC to compute the geometric constraint $({}^{I_k^r}_{I_t}\hat{\mathbf{R}}, {}^{I_k^r}\hat{\mathbf{t}}_{I_t})$ between I_t and I_k^r, as well as the set of features whose reprojection error [19] is within a threshold ε_1 (the error tolerance for outlier rejection [15]). At this point, we require that the condition $e_1 : |S_{5pt}| \geq 25$ (i.e., the number of 5pt inliers is no less than 25; see [14] for a probabilistic justification) is satisfied in order to proceed to Step 3; else the system transitions to state ℓ_2 (see Fig. 3).

In Step 3, we distinguish between the states ℓ_0 and ℓ_1. Specifically, when the baseline is short (i.e., ${}^{I_k^r}d_{I_t} \ll {}^{I_t}d_f, {}^{I_k^r}d_f \Leftrightarrow {}^{I_k^r}\mathbf{b}_f \simeq {}^{I_k^r}_{I_t}\mathbf{R}\,{}^{I_t}\mathbf{b}_f$), the 5 degrees of freedom (dof) epipolar constraint:

$$ {}^{I_k^r}\mathbf{b}_f^T \lfloor {}^{I_k^r}\mathbf{t}_{I_t} \times \rfloor {}^{I_k^r}_{I_t}\mathbf{R}\,{}^{I_t}\mathbf{b}_f = 0 \tag{1} $$

degenerates into a 3 dof, rotation-only constraint that is satisfied by all the 5pt inliers. Our algorithm uses this observation to determine if there is sufficient baseline between the current, I_t, and reference, I_k^r, images. In particular, we employ the 2pt RANSAC on the features $f \in S_{5pt}$ to compute the rotation ${}^{I_k^r}_{I_t}\breve{\mathbf{R}}$ between two images and determine $S_{2pt} = \{f \in S_{5pt} \mid 1 - {}^{I_k^r}\mathbf{b}_f^T \, {}^{I_k^r}_{I_t}\breve{\mathbf{R}} \, {}^{I_t}\mathbf{b}_f < \epsilon_2\}$, which is the subset of 5pt inliers that are also 2pt inliers. Lastly, and in order to compensate for the noise in the measurements and the randomness of RANSAC, instead of requiring $|S_{2pt}| = |S_{5pt}|$, we employ the condition $e_2 : \frac{|S_{2pt}|}{|S_{5pt}|} > 0.94$ to declare small baseline (i.e., state ℓ_1).

Depending on the state of our system (ℓ_0, ℓ_1, or ℓ_2), in what follows, we describe the process for controlling the quadrotor.

3.2.2 Wide Baseline (ℓ_0)

Improving the Motion Estimate

In practice, when the quadrotor navigates through long corridors or open spaces, S_f may contain features at various depths, some of which, typically the faraway ones, may negatively affect the motion estimate. Note that such features, satisfy the 2pt RANSAC. To remove them, we define as $S'_{5pt} = S_{5pt} \setminus S_{2pt}$, run again the 5pt RANSAC on the features $f \in S'_{5pt}$, and use the winning hypothesis to initialize an iterative batch-least squares algorithm [22] to improve the accuracy of the estimated desired motion between I_t and I_k^r.

At this point, we note that although the desired motion between I_t and I_k^r may comprise 5 dof (3 for the relative roll, pitch, yaw and 2 for the unit vector, $\mathbf{t}$, of translation), given the kinematic and actuation constraints of the quadrotor (e.g., it cannot achieve non-zero roll or pitch angle while staying still), our controller seeks to match the desired motion only along 3 dof: The t_x, t_y projection of the desired unit vector, $\mathbf{t}$, of translation on the horizontal plane,[4] and the desired (relative) yaw angle ${}^{I_k^r}\hat{\psi}_{I_t}$. Moreover, and in order to maintain an almost constant-velocity flight, we scale t_x and t_y by v_0 (the maximum velocity that the optical-flow sensor can measure) and provide our controller with the following desired motion vector:

$$\mathbf{r} = \begin{bmatrix} v_x^d \\ v_y^d \\ \hat{\psi} \end{bmatrix} = \begin{bmatrix} t_x \, v_0 \\ t_y \, v_0 \\ {}^{I_k^r}\hat{\psi}_{I_t} \end{bmatrix} \tag{2}$$

Note that this desired motion vector will need to be appropriately modified in the presence of obstacles (see Sect. 3.2.5).

[4]Note that since all images were recorded at about the same height, the z component of the desired motion estimate is rather small after the first reference image and we subsequently ignore it. Instead, we use the distance-to-the-ground measurements to maintain a constant-altitude flight.

Controller

In order to determine the control input, $\mathbf{u}_k(t)$ (roll, pitch, yaw-rate, and thrust), that we must provide to the quadrotor's attitude controller so as to achieve the desired velocity, we employ the vehicle's kinematic equations, linearized about the equilibrium (near hover condition - see [14]):

$$\begin{bmatrix} \dot{v}_x(t) \\ \dot{v}_y(t) \end{bmatrix} = g \begin{bmatrix} \theta(t) \\ -\phi(t) \end{bmatrix} \tag{3}$$

$$\ddot{z}(t) = \frac{1}{m}\tau(t) - g \tag{4}$$

where g is the gravity, m is the quadrotor's mass, and $\phi(t), \theta(t)$, and $\tau(t)$ are the roll, pitch, and thrust of the quadrotor in ego-centric coordinates, respectively.

To compute the velocity error, we use the estimates, $\hat{v}_x$, $\hat{v}_y$, from the optical-flow sensor, to form:

$$\begin{bmatrix} e_{v_x}(t) \\ e_{v_y}(t) \end{bmatrix} = \begin{bmatrix} v_x^d(t) - \hat{v}_x(t) \\ v_y^d(t) - \hat{v}_y(t) \end{bmatrix} \tag{5}$$

Furthermore, the height error, e_z, is defined as the difference between the desired altitude and the estimated height $\hat{z}$ from the downward-pointing ultrasonic sensor:

$$e_z(t) = z^d(t) - \hat{z}(t) \tag{6}$$

Lastly, based on (4), (5), (6) and $\hat{\psi}$ in (2), we form a PID controller that computes the desired control input to the system as:

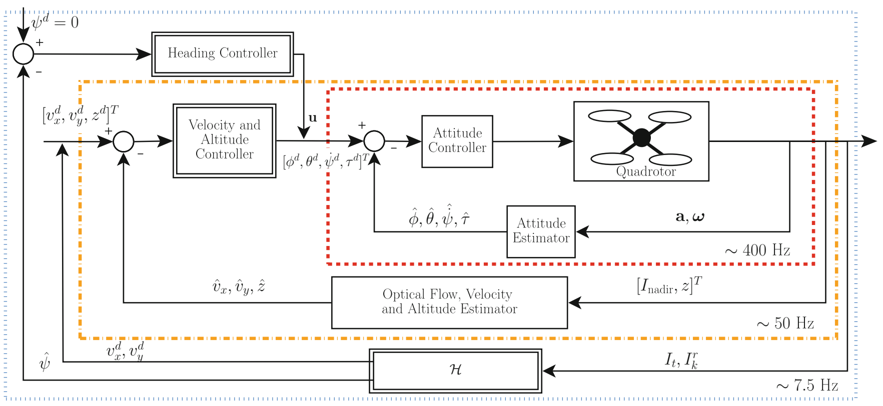
$$\mathbf{u}_k(t) = \begin{bmatrix} \theta^d(t) \\ \phi^d(t) \\ \tau^d(t) \\ \dot{\psi}^d(t) \end{bmatrix} = \begin{bmatrix} k_{p,v_x} e_{v_x}(t) + k_{i,v_x} \int e_{v_x}(t)dt \\ -k_{p,v_y} e_{v_y}(t) - k_{i,v_y} \int e_{v_y}(t)dt \\ k_{p,z} e_z(t) + k_{i,z} \int e_z(t)dt + k_{d,z}\dot{e}_z(t) \\ -k_{p,\psi}\hat{\psi} \end{bmatrix} \tag{7}$$

The gains k_p, k_i, and k_d that ensure high response, zero tracking error, and robustness were found as described in [16].

Figure 4, describes the 3-control-loop implementation of our algorithm on the DJI F450 quadrotor. The outer loop runs at 7.5 Hz and determines the desired 2D velocity, v_x, v_y, and the yaw $\hat{\psi}$ (see Sect. 3.2.2). The desired velocity and height control loop (middle loop) runs at 50 Hz and provides the roll, pitch, and thrust setpoints to the attitude controller (see [14] for more details).

3.2.3 Short Baseline (ℓ_1)

In case of short baseline, we detect if there is any rotational motion needed to minimize the relative yaw, ${}^{I_k^r}\psi_{I_t}$, between I_t, and I_k^r. To do so, we first improve the



$\psi^d = 0$
Heading Controller
$[v_x^d, v_y^d, z^d]^T$
Velocity and Altitude Controller
$\mathbf{u}$
$[\phi^d, \theta^d, \dot{\psi}^d, \tau^d]^T$
Attitude Controller
Quadrotor
$\hat{\phi}, \hat{\theta}, \hat{\dot{\psi}}, \hat{\tau}$
Attitude Estimator
$\mathbf{a}, \boldsymbol{\omega}$
~ 400 Hz
$\hat{v}_x, \hat{v}_y, \hat{z}$
Optical Flow, Velocity and Altitude Estimator
$[I_{\text{nadir}}, z]^T$
~ 50 Hz
v_x^d, v_y^d
$\hat{\psi}$
$\mathcal{H}$
I_t, I_k^r
~ 7.5 Hz

Fig. 4 System block diagram. The *double-line* blocks denote components of our algorithm described in Sects. 3.2.1 ($\mathscr{H}$) and 3.2.2 (Controller)

rotation matrix estimate, ${}^{I_k^r}_{I_t}\breve{\mathbf{R}}$, by employing the least-squares method of [21] on the features $f \in S_{2pt}$ using as initial estimate, the one from the minimal solver of the 2pt RANSAC. After extracting the yaw component, if $|{}^{I_k^r}\breve{\psi}_{I_t}| > \tau_3$,[5] we send the desired rotation-in-place motion $\mathbf{r}^T = [0 \;\; 0 \;\; {}^{I_k^r}\breve{\psi}_{I_t}]^T$ to the controller to minimize the relative yaw between I_t, and I_k^r; else we switch to the next reference image in the path $\mathscr{P}$.

Note that when we have direct access to the attitude estimator, as in the case of the Bebop, we can leverage the yaw angle (computed off-line - see Sect. 3.1.2) between the first and last rotation-only reference images to speed up the execution of this path segment. Specifically, the precomputed relative yaw angle is provided to the controller to perform a "blind" rotation in place. Once this is complete, the quadrotor queries the VT to confirm that the last rotation-only reference image has been reached or determine the remaining rotation between the current image and the last rotation-only reference image.

3.2.4 Lost Mode (ℓ_2)

When the quadrotor loses track of the current reference image, we refer to the last reference image where it computed good matches as I^r_{lost}. There are four possible scenarios that can cause the quadrotor to get lost:

- The robot enters a featureless region.
- The robot enters a region where the result from the FREAK feature matching between I_t and I_k^r is unreliable.

[5]This threshold depends on the onboard camera's fov and is selected so as to ensure a significant overlap (more than 80%) between the current camera image and the next reference image.

- The quadrotor significantly deviates from its current path, in order to avoid an obstacle.
- Dynamic obstacles (e.g., people) obstruct the quadrotor's view or path.

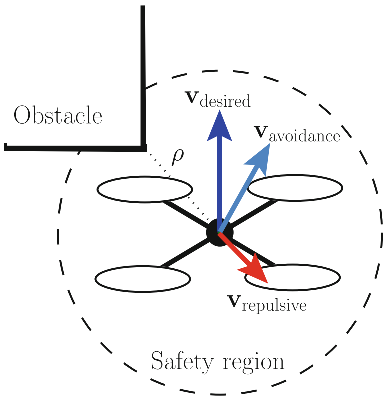
Our recovery method is as follows: While hovering, the quadrotor queries the VT with I_t and successively evaluates among the returned images to find the one that has at least 35 features in common with I_t that pass the 5pt RANSAC. If the above search fails for the top 10 images, the quadrotor switches to a "blind" motion strategy following the same type of motion as before it was lost (i.e., translation or rotation based on I^r_{lost}) for 0.5 s and then attempts again to retrieve a good reference image I^r_{best}. This iterative process is repeated for 10 times before declaring that the quadrotor is lost, in which case, it autonomously lands.

3.2.5 Obstacle Detection and Avoidance

To avoid collisions while following the reference path, we combine the desired motion from the PBVS algorithm with a "repulsive" velocity defined using the ultrasonic sensors. Specifically. we denote as $\rho = 1$ m the radius of the safety region centered around the quadrotor, $o_k(t)$ the measurement of the k^{th} ultrasonic sensor, and $\boldsymbol{\xi}_k$ the 2D unit vector of direction of this ultrasonic sensor relative to the quadrotor. Let γ be a constant relative-distance-to-velocity gain, we then construct a 2D repulsive velocity vector as:

$$\mathbf{v}^k_{\text{repulsive}}(t) = \begin{cases} \mathbf{0} & \text{if } o_k(t) \geq \rho \\ -\gamma * (\rho - o_k(t)) * \boldsymbol{\xi}_k & \text{if } o_k(t) < \rho \end{cases}$$

and set as (Fig. 5): $\mathbf{v}_{\text{avoidance}}(t) = \mathbf{v}_{\text{desired}}(t) + \sum_{k=1}^{8} \mathbf{v}^k_{\text{repulsive}}(t)$



$\mathbf{v}_{\text{desired}}$
$\mathbf{v}_{\text{avoidance}}$
Obstacle
ρ
$\mathbf{v}_{\text{repulsive}}$
Safety region

Fig. 5 Schematic of the velocity determination for obstacle avoidance

4 Experimental Results

We performed experiments with two quadrotors (the Parrot Bebop and the DJI F450) in two different scenarios: Firstly, inside a motion-capture room to evaluate the accuracy of our approach using ground truth from a VICON motion-capture system [28]. Then, in a large indoor area to evaluate the algorithm's performance under challenging conditions.

4.1 System Setup

The DJI F450 quadrotor is equipped with a NAZA attitude controller, a PX4Flow [20] for height and velocity measurements, 8 MaxBotix ultrasonic range finders, and Google's Tango smartphone. The phone has a built-in 180° fisheye camera and a quad-core ARM processor. Note that we do not use the built-in estimator of this device. The quadrotor also carries: (i) An Arduino microcontroller to generate the signals required to operate the quadrotor and switch between manual and autonomous mode, (ii) An ODroid-U3 ARM-based computer used for allowing the cell phone and the Arduino to communicate, and (iii) A wireless router for debugging during test flights. It should be pointed out that the ODroid-U3 and the router are only used for debugging purposes; all computations are performed on the smartphone.

The Bebop, on the other hand, carries a MEMS IMU, a downward-pointing Aptina MT9V117 camera used for optical flow, an ultrasonic sensor for measuring the distance to the ground, and a forward-pointing Aptina MT9V002 camera which we use for visual navigation. All processing is carried onboard Bebop's ARM Cortex A9 800 MHz dual-core processor.

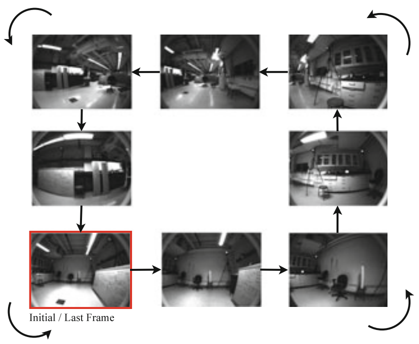
4.2 Short Experiment with Ground Truth

In this experiment, we recorded an image sequence describing a rectangular path (approximately 2×2.5 m) within the motion-capture room while recording the ground truth for the phone's camera motion. The path was created such that the poses of the initial and the last frame coincide. Then, we ran our PBVS algorithm three times on the image sequence while recording the quadrotor's pose with the VICON. Figure 6 (left) shows a subset of the reference camera images along the path, while Fig. 6 (right) depicts the reference trajectory, the location of the reference images of Fig. 6 (left), and the path followed by the quadrotor. As evident, the error between the actual and the reference trajectories is typically within ± 0.5 m. Similar performance was achieved when using the Bebop quadrotor.

4.3 Long Experiments

These experiments took place in the Walter Library's basement which is a challenging environment with numerous specular reflections on the floor, where the quadrotors had to follow a 75 m long path comprising translational motion segments through open spaces as well as rotations in place in order to navigate through narrow passages. Some of these maneuvers were in front of areas where most of the visible scene was behind a coffee-shop glass front whose reflections posed a significant challenge to the motion-estimation algorithm. Figure 7 shows the blueprint of the experimental area depicting the reference visual path (red bold line), and snapshots of two quadrotors in flight.

During the experiment, the Bebop was able to complete the reference trajectory in 240 s, at an average speed of 40 cm/s, despite getting lost and recovering its path



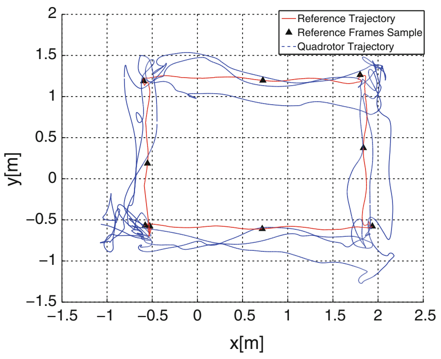


Fig. 6 Short experiment with ground truth: (*left*) Sample of camera frames used as reference images; (*right*) Comparison between the reference trajectory and the actual quadrotor trajectory

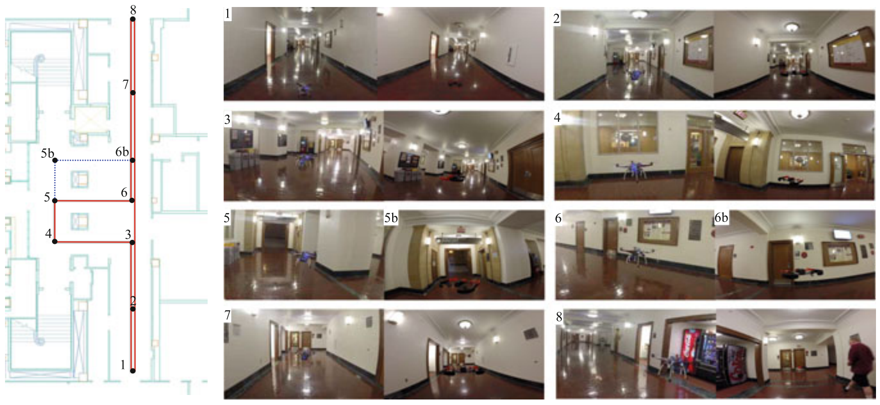


Fig. 7 Long experiment: Blueprint of the experimental area, reference path for the DJI F450 (1-4-5-6-7-8-1) and the Parrot Bebop (1-4-5b-6b-7-8-1), and snapshots of both quadrotors along their paths (1–8)

twice. On the other hand, it took the F450 quadrotor 450 s (average speed of 20 cm/s) to complete approximately the same path after getting lost 3 times. The difference in the performance of the two quadrotors is mainly due to the lower maximum-velocity sensing capabilities of the PX4Flow sensor onboard the F450 but also because of the agility and safety (no sonars were required) that the smaller size Bebop quadrotor provided. The videos for the Bebop and F450 experiments are available at [1].

5 Conclusion and Future Work

In this paper, we presented a visual-servoing algorithm that allows quadrotors to autonomously navigate within a previously-mapped area. In our work, the map is constructed offline from images collected by a user walking though the area of interest and carrying a cell phone or a quadrotor. Specifically, and in order to increase efficiency, the visual map is represented as a graph of images linked with edges whose weights (cost to traverse) are inversely proportional to the number of features common to them. Once the visual graph is constructed, and given as input the start and goal location of the quadrotor, it automatically determines the desired path as a sequence of reference images. This information is provided to the quadrotor, which estimates in real time the motion that minimizes the difference between its current and reference images, and controls its roll, pitch, yaw-rate, and thrust for achieving that.

Besides the ease of path-specification, a key advantage of our approach is that by employing a mixture of 2 and 5pt RANSAC for determining the type of motion required (rotational, translational with close-by or faraway scene), and for selecting, on the fly, the next reference image, it is able to navigate through areas comprising featureless corridors, as well as narrow passages. Moreover, it is able to cope with static and moving obstacles and, in many cases, recover its path after losing track of its reference image. The accuracy of the proposed algorithm was assessed using motion-capture data within a small area, while its robustness to lighting conditions and in-place rotations was demonstrated by autonomously navigating along a 75 m path through a large building. Lastly, and as part of our ongoing work, we are currently extending our algorithm to combine visual and inertial measurements [18] that will improve the accuracy of the velocity estimates, and, thus, allow us to further increase the quadrotors' operational speed.

6 Appendix

6.1 Singular Condition of the 5pt RANSAC Minimal Solver

Consider a feature f appearing both in I_t and I_k^r which satisfies the following geometric constraint:

$$^{I_k^r}d_f \, ^{I_k^r}\mathbf{b}_f = {}^{I_t}d_f \, ^{I_k^r}_{I_t}\mathbf{R} \, ^{I_t}\mathbf{b}_f + {}^{I_k^r}d_{I_t} \, ^{I_k^r}\mathbf{t}_{I_t} \tag{8}$$

Note that when sufficient baseline exists between the current and reference images, (8) can be projected on the normal to the epipolar plane to yield the epipolar constraint

$$^{I_k^r}\mathbf{b}_f^T \lfloor {}^{I_k^r}\mathbf{t}_{I_t} \times \rfloor {}^{I_k^r}_{I_t}\mathbf{R} {}^{I_t}\mathbf{b}_f = 0 \tag{9}$$

By employing five pairs of features that satisfy (9), we can estimate the desired 5 dof motion using the 5pt RANSAC algorithm [26].

On the other hand, consider the case when the relative position between I_t and I_k^r is significantly smaller compared to either distance to the feature f. Without loss of generality, we assume that $^{I_k^r}d_{I_t} \ll {}^{I_t}d_f \Rightarrow \frac{^{I_k^r}d_{I_t}}{^{I_t}d_f} \simeq 0$ and employ the law of cosines in the triangle defined by the focal points of the two cameras and f:

$$^{I_k^r}d_f^2 + {}^{I_t}d_f^2 - 2 {}^{I_k^r}d_f {}^{I_t}d_f cos(\gamma) = {}^{I_k^r}d_{I_t}^2 \Rightarrow$$
$$\left(\frac{^{I_k^r}d_f}{^{I_t}d_f}\right)^2 + 1 - 2\frac{^{I_k^r}d_f}{^{I_t}d_f}cos(\gamma) = \left(\frac{^{I_k^r}d_{I_t}}{^{I_t}d_f}\right)^2 \simeq 0 \Rightarrow \left(\frac{^{I_k^r}d_f}{^{I_t}d_f} - 1\right)^2 + 2\frac{^{I_k^r}d_f}{^{I_t}d_f}(1 - cos(\gamma)) \simeq 0$$

which implies that $^{I_k^r}d_f \simeq {}^{I_t}d_f = d$. Dividing both sides of (8) with d and considering that $^{I_k^r}d_{I_t} \ll d$, yields

$$^{I_k^r}\mathbf{b}_f \simeq {}^{I_k^r}_{I_t}\mathbf{R} {}^{I_t}\mathbf{b}_f \tag{10}$$

Therefore, given two pairs of inliers, the rotation matrix $^{I_k^r}_{I_t}\mathbf{R}$ in (10) can be found in closed form (see Sect. 6.2).

6.2 2pt RANSAC Minimal Solver

Consider bearing measurements $^{I_1}\mathbf{b}_{f_1}$, $^{I_1}\mathbf{b}_{f_2}$, $^{I_2}\mathbf{b}_{f_1}$, $^{I_2}\mathbf{b}_{f_2}$ to two features from two images, and assume that the motion between them is purely rotational, thus

$$^{I_2}\mathbf{b}_{f_i} = \mathbf{R}({}^{I_2}_{I_1}\bar{q}) {}^{I_1}\mathbf{b}_{f_i}, i = 1, 2$$

where $^{I_2}_{I_1}\bar{q}$ is the unit quaternion of rotation.

If $({}^{I_2}\mathbf{b}_{f_1} - {}^{I_1}\mathbf{b}_{f_1}) \times ({}^{I_2}\mathbf{b}_{f_2} - {}^{I_1}\mathbf{b}_{f_2}) \neq \mathbf{0}$, then the closed-form solution is:

$$^{I_2}_{I_1}\bar{q} = \gamma \begin{bmatrix} ({}^{I_2}\mathbf{b}_{f_1} - {}^{I_1}\mathbf{b}_{f_1}) \times ({}^{I_2}\mathbf{b}_{f_2} - {}^{I_1}\mathbf{b}_{f_2}) \\ ({}^{I_2}\mathbf{b}_{f_2} - {}^{I_1}\mathbf{b}_{f_2})^T ({}^{I_2}\mathbf{b}_{f_1} + {}^{I_1}\mathbf{b}_{f_1}) \end{bmatrix}$$

else

$$ {}^{I_2}_{I_1}\bar{q} = \eta \begin{bmatrix} ({}^{I_2}\mathbf{b}_{f_1} \times {}^{I_2}\mathbf{b}_{f_2}) \times ({}^{I_1}\mathbf{b}_{f_1} \times {}^{I_1}\mathbf{b}_{f_2}) \\ ({}^{I_2}\mathbf{b}_{f_1} \times {}^{I_2}\mathbf{b}_{f_2})^T ({}^{I_2}\mathbf{b}_{f_1} \times {}^{I_2}\mathbf{b}_{f_2} + {}^{I_1}\mathbf{b}_{f_1} \times {}^{I_1}\mathbf{b}_{f_2}) \end{bmatrix} $$

where γ, η are the normalization factors that ensure unit length.

References

1. Autonomous flights through image-defined paths (videos). http://mars.cs.umn.edu/research/quadrotor_project.php
2. Alahi, A., Ortiz, R., Vandergheynst, P.: FREAK: Fast retina keypoint. In: Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition, Providence, RI, pp. 510–517, 16–21 June 2012
3. Azrad, S., Kendoul, F., Nonami, K.: Visual servoing of quadrotor micro-air vehicle using color-based tracking algorithm. J. Syst. Design Dyn. **4**(2), 255–268 (2010)
4. Bebop Drone. http://www.parrot.com/products/bebop-drone/
5. Bills, C., Chen, J., Saxena, A.: Autonomous MAV flight in indoor environments using single image perspective cues. In: Proceedings of the IEEE International Conference on Robotics and Automation, Shanghai, China, pp. 5776–5783, 9–13 May 2011
6. Bourquardez, O., Mahony, R., Guenard, F.C.N.: Image-based visual servo control of the translation kinematics of a quadrotor aerial vehicle. IEEE Trans. Robot. **25**(3), 743–749 (2009)
7. Chaumette, F., Hutchinson, S.: Visual servo control, part i: basic approaches. IEEE Robot. Autom. Mag. **13**(4), 82–90 (2006)
8. Chaumette, F., Hutchinson, S.: Visual servo control, part ii: advanced approaches. IEEE Robot. Autom. Mag. **14**(1), 109–118 (2007)
9. Chen, Z., Birchfield, R.T.: Qualitative vision-based path following. IEEE Trans. Robot. **25**(3), 749–754 (2009)
10. Cormen, T.H., Leiserson, C.E., Rivest, R.L., Stein, C.: Introduction to Algorithms. MIT Press, Cambridge (2001)
11. Courbon, J., Mezouar, Y., Guenard, N., Martinet, P.: Vision-based navigation of unmanned aerial vehicle. Control Eng. Pract. **18**(7), 789–799 (2010)
12. Courbon, J., Mezouar, Y., Martinet, P.: Indoor navigation of a non-holonomic mobile robot using a visual memory. Auton. Robot. **25**(3), 253–266 (2008)
13. Diosi, A., Šegvić, S., Remazeilles, A., Chaumette, F.: Experimental evaluation of autonomous driving based on visual memory and image-based visual servoing. IEEE Trans. Robot. **12**(3), 870–883 (2011)
14. Do, T., Carrillo-Arce, L.C., Roumeliotis, S.I.: Autonomous flights through image-defined paths, Technical Report (2015). http://mars.cs.umn.edu/publications.html
15. Fischler, M., Bolles, R.: Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. Commun. ACM **24**(6), 381–395 (1981)
16. Franklin, G.F., Powell, J.D., Workman, M.L.: Digital Control of Dynamic Systems. Addison-Wesley, Reading (1997)
17. Goedemé, T., Tuytelaars, T., Gool, L.V., Vanacker, G., Nuttin, M.: Feature based omnidirectional sparse visual path following. In: Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems, Alberta, Canada, pp. 1806–1811, 2–6 August 2005
18. Guo, C.X., Kottas, D.G., DuToit, R.C., Ahmed, A., Li, R., Roumeliotis, S.I.: Efficient visual-inertial navigation using a rolling-shutter camera with inaccurate timestamps. In: Proceedings of the Robotics: Science and Systems Conference, Berkeley, CA, 12–16 July 2014
19. Hartley, R.I., Zisserman, A.: Multiple View Geometry in Computer Vision, 2nd edn. Cambridge University Press, Cambridge (2004). ISBN: 0521540518

20. Honegger, D., Meier, L., Tanskanen, P., Pollefeys, M.: An open source and open hardware embedded metric optical flow CMOS camera for indoor and outdoor applications. In: Proceedings of the IEEE International Conference on Robotics and Automation, Karlsruhe, Germany, pp. 1736–1741, 6–16 May 2013
21. Horn, B.: Closed-form solution of absolute orientation using unit quaternions. J. Opt. Soc. America A **4**(4), 629–642 (1987)
22. Horn, B.: Relative orientation. Int. J. Comput. Vis. **4**(1), 59–78 (1990)
23. Lee, D., Ryan, T., Kim, H.J.: Autonomous landing of a VTOL UAV on a moving platform using image-based visual servoing. In: Proceedings of the IEEE International Conference on Robotics and Automation, Saint Paul, MN, pp. 971–976, 14–18 May 2012
24. Mariottini, G.L., Roumeliotis, S.I.: Active vision-based robot localization and navigation in a visual memory. In: Proceedings of the IEEE International Conference on Robotics and Automation, Shanghai, China, pp. 6192–6198, 9–13 May 2011
25. Nguyen, T., Mann, G.K.I., Gosine, R.G.: Vision-based qualitative path-following control of quadrotor aerial vehicle. In: Proceedings of the IEEE International Conference on Unmanned Aircraft Systems, Orlando, FL, pp. 412–417, 27–30 May 2014
26. Nistér, D.: An efficient solution to the five-point relative pose problem. IEEE Trans. Patter Anal. Mach. Intell. **26**(6), 756–770 (2004)
27. Nistér, D., Stewénius, H.: Scalable recognition with a vocabulary tree. In: Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition, New York, NY, pp. 2161–2168, 17–22 June 2006
28. Vicon Motion Systems Ltd. http://www.vicon.com/

Altitude Estimation and Control of an Insect-Scale Robot with an Onboard Proximity Sensor

E. Farrell Helbling, Sawyer B. Fuller and Robert J. Wood

1 Introduction

Insect-scale MAVs (50–500 mg) are envisioned for many applications, including hazardous environment exploration and assisted agriculture. However, flight at this scale poses a control challenge for stability, as rotational acceleration increases with a decrease in the vehicle's characteristic length, scaling as l^{-1} [15]. The vehicle's flight dynamics are also unstable, requiring the flight controller to perform continuous corrective maneuvers [1]. Therefore, in addition to meeting the stringent mass and power requirements for vehicles at this scale, onboard sensors must also provide low latency, high bandwidth information to the active controller.

The Harvard RoboBee (see Fig. 1) is the first MAV under 100 mg to lift its own weight and demonstrate controlled flight under external power [16]; however, an external, camera-based motion capture system (Vicon T040 System, Oxford UK) was used to estimate the position and orientation of the robot. For the RoboBee to operate in unstructured environments, it must be equipped with sensors that can estimate the vehicle's state to stabilize the dynamics and sense its external environment. Insects are a key source of inspiration for flight control, they rely on a multitude of sensory organs that provide estimates of relevant state variables to use in combination with reflexive or high-level control architectures to adjust flight accordingly [20].

Vision has been shown to be particularly important for navigation at small scales; GPS is too imprecise, with an accuracy of 1–10 m, and currently available emissive sensors such as radar, scanning laser rangefinders, and sonar do not meet the weight requirements (∼40 mg) of the vehicle [4]. MAVs have adopted vision sensors for navigation at scales one to two orders of magnitude larger than the RoboBee,

E.F. Helbling (✉) · S.B. Fuller · R.J. Wood
The Wyss Institute for Biologically-Inspired Engineering, Harvard University
John A. Paulson School of Engineering and Applied Sciences,
60 Oxford Street, Cambridge, MA 02138, USA
e-mail: ehelbling@seas.harvard.edu

S.B. Fuller
Department of Mechanical Engineering, University of Washington,
Seattle, WA 98195, USA

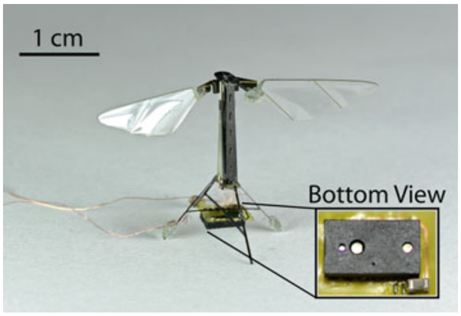


Fig. 1 The Harvard RoboBee is an 80 mg flapping-wing MAV that has demonstrated controlled flight using external motion capture cameras and reflective markers on the base and nose of the robot. Here we incorporate an infrared (IR) time-of-flight (ToF) sensor mounted on the base of the robot and use it to detect and control distance from the ground during flight. Inset: VL6180x (ST Microelectronics) mounted on a custom flex circuit. The sensor is 5 mm in length

including [3, 7]. Other researchers have demonstrated high bandwidth, low latency, and lightweight biomimetic optical sensors with the goal of stabilizing many aspects of the RoboBee's flight, including altitude on fixed guide wires [9], and orientation after takeoff [11].

In addition to these vision-based sensors, inertial measurement units (IMUs) – a system consisting of gyroscopes, accelerometers, and magnetometers – have long been used for flight control and stabilization of larger aircraft. Demand for miniaturization by the consumer electronics industry has resulted in sensors that now meet the mass ($\sim$10–100 mg) and power ($\sim$10 mW) requirements of the RoboBee and have been integrated onto the vehicle. These sensors demonstrated low-latency feedback and sufficient noise rejection, controlling motion about the pitch and yaw axes [12], as well as providing flight stabilization at takeoff and hovering [10]. In addition to stabilizing the attitude of the vehicle, the hovering controller from [6] requires estimates of rotation rate about the body axes, the absolute orientation of the vehicle, the lateral velocity and position, as well as the altitude to hover with asymptotic stability.

In addition to facilitating altitude regulation, an altitude estimate, when combined with other sensors, can be used to estimate other aspects of the vehicle's motion. Future research into autonomous, visually-guided flight at this scale will require accurate altitude estimation. An onboard, downward-facing optic flow sensor coupled with an absolute altitude estimate can provide the vehicle's forward velocity, as in [13] and demonstrated by landing honeybees in nature [20].

Here we consider a millimeter-scale infrared (IR) time-of-flight (ToF) sensor to measure the absolute position of the vehicle from the ground. We demonstrate that the sensor meets the mass and power requirements of the vehicle while providing feedback at a sufficient rate ($\sim$50 Hz) to control the altitude of the vehicle during hovering flight.

This paper describes the altitude dynamics of the RoboBee and estimates the minimum feedback rate necessary to stabilize this degree of freedom (Sect. 2);

discusses the sensors that meet the mass, power, and frequency requirements as well as describes the integration and calibration of the ToF sensor onboard the vehicle (Sect. 3); and demonstrates controlled flight of the vehicle with onboard altitude estimates from the sensor at multiple setpoints (Sect. 4).

2 Altitude Dynamics

The RoboBee used in these experiments was first presented in [17]. Piezoelectric actuators drive each of the wings, and the system has been shown to produce lift and body torques to take off, land, hover and perform aggressive maneuvers [6]. The hovering controller is composed of three sub-controllers – the lateral controller, the attitude controller, and the altitude controller. In this work, we eliminate the need for an altitude estimate from Vicon by providing it instead from an onboard sensor (see Fig. 2).

The altitude controller assumes that the robot is always in the upright orientation – an assumption that is maintained by the attitude controller. This requires that the thrust vector is always aligned with the $z-$axis, along the same axis as the gravitational force. Assuming that the robot maintains an attitude that is nearly upright, the controller can be designed around a linearization of its dynamics at hover, neglecting second-order effects arising from perturbations from zero attitude.

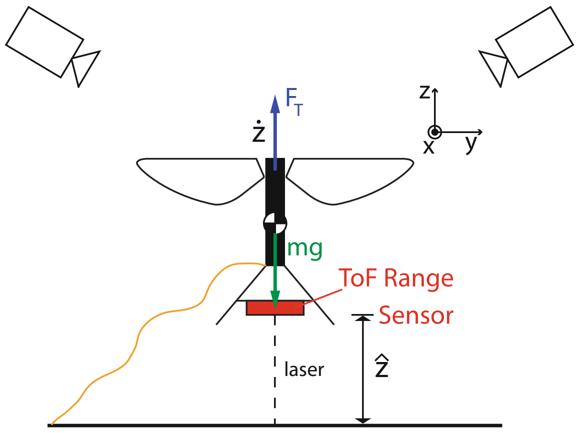


Fig. 2 Model of the altitude dynamics of the RoboBee and diagram of the flight apparatus. The altitude controller assumes upright orientation, reducing the model to a single dimension, z. The thrust force, F_T, generated by the flapping wings acts along the body $z-$axis, with the gravitational force of the body acting at the center of mass. The robot moves with velocity, $\dot{z}$, along the $z-$axis. The sensor (mounted below the robot) estimates the distance of the robot from the ground, $\hat{z}$, with Vicon cameras estimating the lateral position and attitude during flight. The robot is tethered for power and control signals, as well as sensor communications

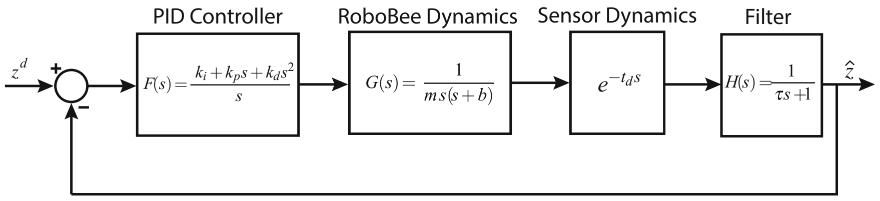

PID Controller
RoboBee Dynamics
Sensor Dynamics
Filter
z^d
+
−
$F(s) = \frac{k_i + k_p s + k_d s^2}{s}$
$G(s) = \frac{1}{ms(s+b)}$
$e^{-t_d s}$
$H(s) = \frac{1}{\tau s + 1}$
$\hat{z}$

Fig. 3 The closed-loop dynamic model of the robot and sensor for perturbations away from the hovering setpoint. The PID controller (gains k_p, k_i, k_d) computes a thrust force F_T that minimizes error between the desired setpoint z^d and the measured altitude $\hat{z}$. This thrust force acts as an input to the dynamics of the robot (Eq. 1), forcing the robot to a new altitude. The sensor reads this new altitude with some time delay t_d and this response is filtered with a low pass filter with cutoff frequency $f_c = 20\,\text{Hz}$

An understanding of the vehicle dynamics along the body $z-$axis is required to determine the maximum latency (sensor time delay) permissible for altitude control. The altitude dynamics can be described by a linear, second-order system:

$$\ddot{z} = \frac{F_T}{m} - b\dot{z} - g \ , \tag{1}$$

where F_T is the thrust force, m is the mass of the robot ($m = 110 \times 10^{-6}\,\text{kg}$), b is the damping constant ($b = 1.2\,\text{s}^{-1}$ [5]), and g is the gravitational acceleration.

To determine the maximum latency in sensor estimates, we computed the closed loop poles of the system and determined the maximum settling time. The altitude controller is a proportional-integral-derivative (PID) controller that reduces error between a desired setpoint and the measured height of the robot. A feedforward term in the controller balances the gravitational force, and thus the input is the thrust force. The sensor latency is modeled using a second-order Padé approximation of pure time delay, and the raw sensor data is processed with a low-pass filter ($f_c = 20\,\text{Hz}$). We simulated multiple time delays in the range $t_d = 0.01 - 0.5\,\text{s}$ (see Fig. 3).

As we are focused on determining the relationship between sensor latency and the settling time of the system, we used the Ziegler–Nichols method [2] to determine the controller gains of the system in order to respond to the changes in latency in a consistent manner.

Figure 4 displays the settling time of the system plotted against various time delays of sensor estimates for the feedback system shown in Fig. 3. As expected, as the time delay of the sensor increases, the settling time also increases. It is interesting to note that the rate of increase in settling time decreases with sensor latency, suggesting that sensor latency will decrease system performance but does not lead to instability with proper tuning of the controller. With this relationship, we are able to determine the minimum sensor time delay for maneuvers that require specific settling times and analyze the tradeoffs between system performance and power and computation costs. For our current applications, a 2.5 s settling time is adequate and therefore sensor latency less than 0.5 s is reasonable.

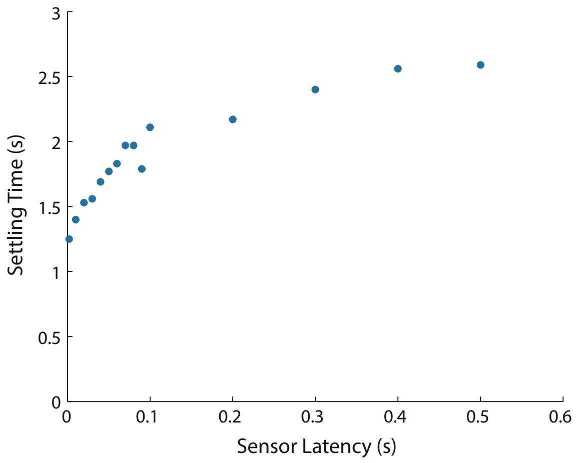


Fig. 4 The system settling time with respect to sensor latency. Modeling the closed-loop system in Fig. 3, we calculated the closed loop poles of the system to determine settling time with various sensor dynamics. The controller gains at each trial were computed with the Ziegler–Nichols method and the phase margin is $\phi_m = 60°$ for all trials

3 Sensor Selection and Integration

The current vehicle has severe restrictions on payload mass and onboard power consumption. While [17] measured lift force sufficient to carry an additional 70 mg payload, the maximum payload carried by the robot during hovering flight was 40 mg, to allow for additional control authority in [10].

While the vehicle does not currently carry an onboard power supply, minimizing the power consumed by sensing is a major consideration for future applications. The vehicle consumes 19 mW of power during flight [16], consistent with similarly sized insects [8]. We are allotting 10 mW towards sensing and computation, based on power requirements of current high performance sensors.

Table 1 Altitude sensors for Centimeter-Scale Flapping-Wing MAVs

Sensors	Mass (mg)	Power (mW)	Frequency	Remarks
Accelerometer [10]	24	1	1 kHz	Integration drift (2 m in 1 s)
Time-of-flight [21]	20	6	50 Hz	VCSEL light source requires regulator
Optical flow [9]	15	15	40 Hz	Computationally expensive
Pressure [18]	40	3	125 Hz	Insufficient resolution (20 cm)
Sonar [19]	80	10	40 Hz	Does not meet mass constraints
IR range [14]	16	6	10 Hz	Dependent on reflective material

In Table 1, we have tabulated a set of candidate sensors that meet the requirements listed above – accelerometers, IR ToF, optic flow, pressure, sonar, and IR range detectors – that are currently available. Our evaluation is as follows: accelerometers lose accuracy over time due to the drift in sensor estimate after integration; additional velocity or distance sensors would need to be integrated onboard the vehicle to compensate, increasing the sensor payload of the vehicle. Optic flow sensors have demonstrated relative altitude control on the RoboBee [9]. Because these sensors only provide the angular velocity, which depends on both distance and velocity relative to an object, an additional sensor measuring absolute altitude or lateral velocity would need to be added to eliminate steady-state error [13]. Similarly scaled vehicles such as the Delfly [7] have demonstrated controlled altitude with absolute pressure sensors. The small vertical motion available (30 cm) in the flight arena is smaller than the precision of available pressure sensors which have a resolution of approximately 20 cm [18]. Sonar shows promise for altitude estimation, but does not yet meet the mass requirements for this robot. The IR range detector measures the amount of reflected light in the receiver. This sensor meets the mass and power requirements, but has a low feedback rate and is highly dependent on the material off of which the IR light is reflecting. We also found that the sensor signal can saturate in the presence of ambient light. The ToF sensor, while needing an external voltage regulator, has low mass, sufficient bandwidth, and low latency for altitude control in these experiments.

3.1 Time-of-Flight Sensor

As a first step in altitude control, we consider an IR ToF range detector (VL6180x, STMicroelectronics). Time-of-Flight sensors compute distance by measuring the time between the transmission and reception of an IR signal generated by the sensor. This distinguishes them from the more common IR range detectors that measure the amount of reflected light in the receiver. This was an important consideration given that the vehicle's other degrees of freedom are currently sensed using Vicon motion capture cameras that emit IR light.

3.1.1 Integration

To minimize component weight, we made a custom flex circuit using a copper-clad polyimide film (18 μm copper, 12.7 μm kapton) with a capacitor on the power line to help regulate the charge close to the sensor (see Fig. 1 inset). Five 51-gauge copper wires (approximately 0.5 m long) provided power to the sensor and connect the data and clock lines for I2C communication. During operation, the sensor drew approximately 20 mA of current at regular intervals to pulse the IR VCSEL (Vertical-Cavity Surface-Emitting Laser) light source [21]. This amount of current traveling through the wire (approximately 80 Ω/m) caused the voltage at the sensor to drop below operating conditions. We connected a fifth wire to the input voltage line which served as a feedback line to an off board linear voltage regulator (NCP3337, ON Semiconductor). This provided a smooth input voltage to the sensor. The weight of the entire structure was 27 mg. We mounted the sensor on the base of the robot, with the transmitter and receiver directed at the ground (see Fig. 1). This position is close to the center of mass, thus preventing significant moments about any of the body axes. Because the sensor is light-based, there are no interactions between the robot's wing beat frequency and the sensor readings.

An ArduinoMega (ATMega256, Atmel Corporation) communicates with the sensor over I2C at approximately 60 Hz and sends the range response to a computer running xPC Target (Mathworks) through a serial (RS232) connection at 115 kbps. This communication scheme introduces about 20 ms of latency, 15 ms of which are dedicated to the sensor's ranging computation and 5 ms to communication. This produces a feedback rate of approximately 50 Hz, a tenth of the speed of the Vicon system that is currently being used to control the vehicle's altitude, but nominally sufficient according to the calculations in Sect. 2.

3.1.2 Calibration

We calibrated the sensor measurements by manually adjusting the height of the sensor while attached to the robot in the flight arena. We tracked the vehicle's altitude using the Vicon system while simultaneously recording the sensor's output. We made the decision to calibrate the sensor against the current Vicon estimates, as this Vicon system has shown sufficient accuracy for altitude control in previous experiments [6, 17]. Figure 5 displays sensor output (scaled to meters) recorded against the ref-

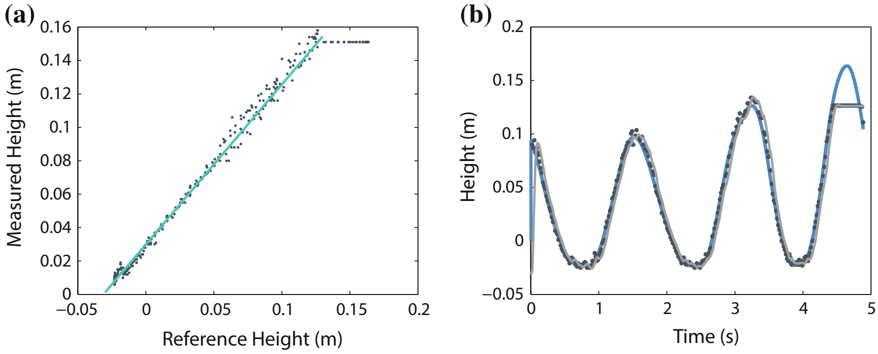


Fig. 5 Sensor Calibration. **a** Sensor measurements (*gray dots*) were taken simultaneously with Vicon measurements (ground truth). A best fit line (teal) was computed: $\hat{z} = 0.96z + 0.030\,\text{mm}$, where the offset is primarily due to the internal calibration of the sensor, which cannot detect distances less than 10 mm, and the calibration of the Vicon arena. **b** Sensor measurements (*gray dots*) plotted on top of the Vicon measurement (*blue line*) with respect to time. The sensor measurements were filtered using a second-order low-pass Butterworth filter, $f_c = 16\,\text{Hz}$ (*grey line*). This filter introduced an 80 ms latency between Vicon and the sensor measurement

erence height for a number of these trials. A line of best fit was calculated for this collection of measurements, $\hat{z} = Az + B$, where $A = 0.96$ and $B = 0.030$ mm. The slope of the line indicates that the sensor is accurately reading changes in altitude at the millimeter scale. The offset compensates for both the sensor's internal offset (the sensor is unable to detect distances less than 10 mm), as well as the calibration of the Vicon system. The estimates also become unreliable close to the surface as well as above 14 cm, providing an operating range for accurate altitude estimation. The error in the sensor estimates increases with height at a roughly constant 3% rate.

Using this linear fit, the sensor measurements were plotted alongside the reference height over a period of approximately six seconds to characterize any drift the sensor may exhibit as well as to determine the effect of the latency on the estimated height. As expected, the estimated measurement aligns with the reference height over all time without any noticeable drift in the sensor. With this data, we also estimate that the sensor measurements have $t_d = 10$ ms of latency compared to the Vicon measurements. This latency increases to approximately $t_d = 80$ ms when the sensor measurements are filtered with a second-order low-pass Butterworth filter ($f_c = 16$ Hz), dictating a settling time of approximately 2 s, based on the results in Fig. 4.

4 Flight Experiments

The robot is not equipped with a power source, computational capability, or a full sensor suite, and is flown inside an controlled flight arena with a volume of approximately 0.3 m $\times$ 0.3 m $\times$ 0.3 m. Six motion capture cameras track the position and orientation of the robot during flight. The control computation is done on the xPC Target at a rate of 5 kHz. The xPC Target commands the power signal through a digital-to-analog converter and high voltage amplifiers. Power is supplied through a tether of four 54-gauge copper wires.

Flight testing begins with open-loop tuning to determine the torque biases about pitch and roll of the robot. To tune the robot, we begin by applying no net torque to the vehicle and observe the trajectory of the vehicle after takeoff. We then apply trim values to oppose the observed torques. This process is repeated until the robot's takeoff is vertically upright.

For all flights mentioned in this section, the robot is attached to a three-filament kevlar thread and suspended above the surface to prevent wear on both the wings and wing hinges during crash landings. These filaments (approximately 30 cm in length) have negligible mass (0.2 mg) compared to that of the robot with the onboard sensor. In addition, we found that the filaments produce negligible torques on the robot during flight. A 10 cm thread cannot support its own weight (20 μg) when extended horizontally, indicating that the bundle cannot exert more than 0.02 μNm of torque on the robot – small compared to the 0.35 μNm of torque produced around the pitch and roll axes during hovering flight.

The altitude controller used during these flight experiments is described in detail in Sect. 2. The hovering flight experiments were done in multiple steps – first the vehicle was commanded to hover using Vicon estimates for altitude to tune the controller gains and provide a baseline flight performance for our results. Altitude was then controlled with estimates from the onboard sensor – four flights were commanded at an altitude of 8 cm to demonstrate repeatability and one flight at an altitude of 10 cm to test the range of the sensor. The attitude and lateral controllers used Vicon estimates of position and orientation for all flights. These are the first demonstrations of controlled altitude with onboard sensing in free flight.

4.1 Vicon Estimates

Once we determined the robot's torque biases, we had to determine the controller gains in subsequent closed loop experiments. To first ensure that the robot had sufficient control authority to hover stably about a setpoint, Vicon was used to provide the estimated altitude measurement. The additional 27 mg at the base of the

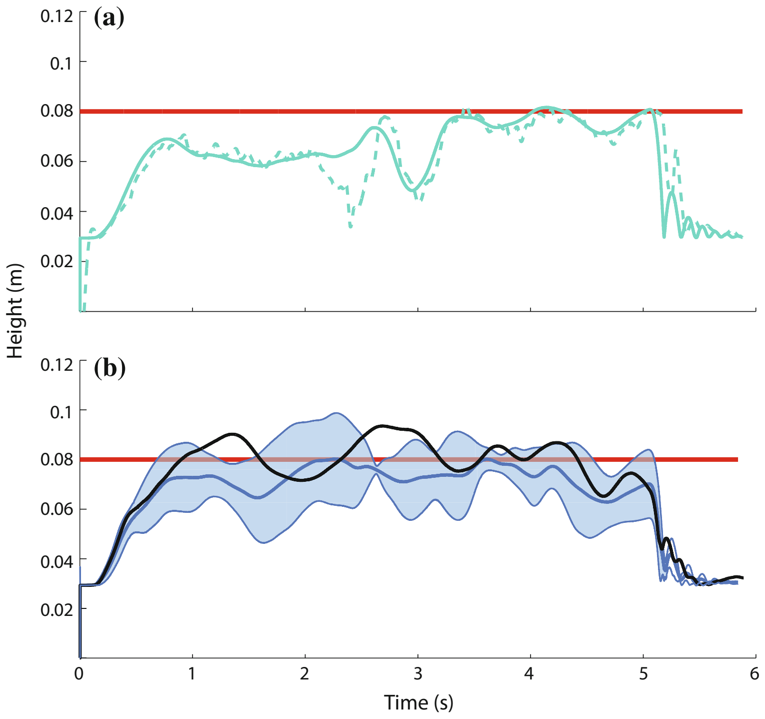


Fig. 6 Closed Loop Flight Experiments. **a** The robot is commanded to reach the setpoint altitude of 8 cm (*red line*) with the sensor providing altitude feedback. The sensor measurement (*dashed*) and the Vicon reference altitude (*solid*) are plotted. **b** The mean and standard deviation of four hovering flight experiments with altitude estimation provided by the sensor are plotted (*blue*). For reference, the baseline hovering flight with altitude estimation from Vicon is plotted (*black*)

robot requires larger thrust forces, and therefore larger flapping amplitudes, to lift off the ground. The black line in Fig. 6 gives an example altitude trajectory of the vehicle during a hovering flight. The vehicle reaches and maintains the altitude setpoint during the 6 s flight (RMS error between the trajectory and the setpoint is 7 mm).

4.2 Sensor Estimates

Having determined that the loaded robot has the control authority to stably hover around a set point, the next step was to determine if the sensor feedback rate was sufficient to control altitude during free-flight. In these flight experiments, the filtered sensor measurements provided input to the altitude controller, while Vicon estimates provided the lateral position and orientation of the robot to stabilize the attitude dynamics during flight. Figure 6a provides an example of a hovering flight with feedback from the ToF sensor. The robot takes off and reaches the setpoint in approximately three seconds. The sensor output tracks the Vicon measurement for the majority of the flight. The largest difference is the sharp downward peak between two and three seconds. This error is most likely due to one of the tethers obstructing the sensor's view of the ground. Three additional trials were performed at this desired setpoint. The mean and standard deviation of these flights can be seen in Fig. 6b alongside the Vicon baseline flight. In these trials, the robot is able to reach the setpoint and sustain hovering flight with an average RMS error of 13.75 mm from the setpoint. We then selected a second altitude setpoint to demonstrate the sensor's ability to provide accurate feedback at other altitudes (see Fig. 7). A sample flight sequence is shown in Fig. 8, where the robot reaches the desired setpoint between 1–2 s and maintains that altitude for the remainder of the flight.

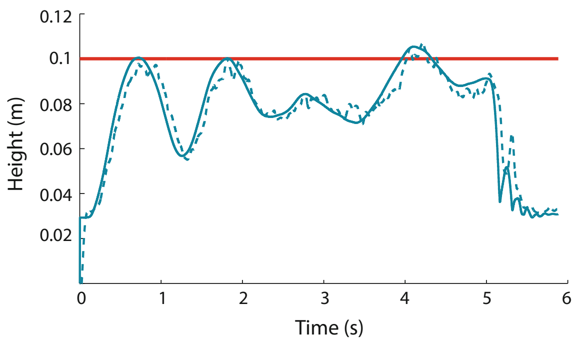


Fig. 7 Sensor Feedback for Hovering Flight at 10 cm altitude. The flight is plotted in time, with the setpoint shown in *red*, sensor data (*dashed line*), and the Vicon data (*solid line*)

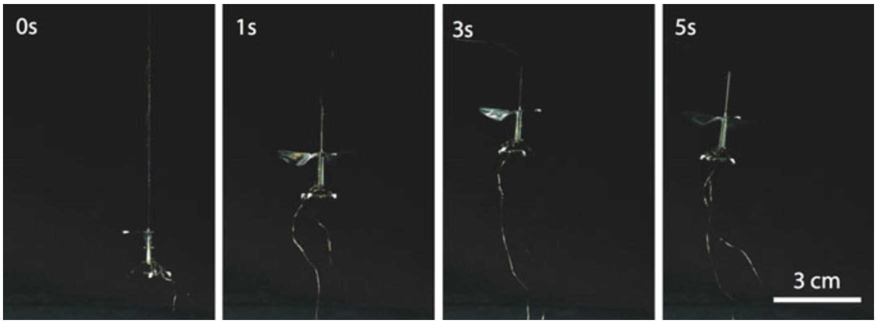
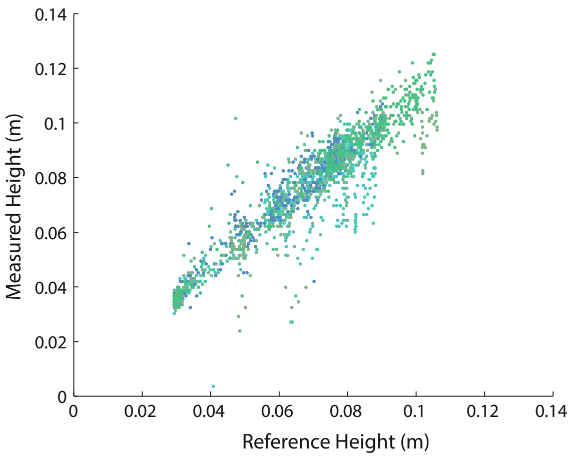


Fig. 8 Sample Flight Sequence. The robot begins at the start altitude of approximately 3 cm, begins flying to the desired altitude at 1 s, reaches the setpoint and maintains that altitude for the remainder of the flight, as demonstrated by the images at 3 and 5 s

Fig. 9 Sensor Calibration Verification. The sensor measurements for all flight experiments are plotted against the Vicon ground truth estimate. The linear trend with unity slope held for all flights, with outliers accounting for less than 5% of all measurements



4.3 *Discussion*

Closed loop flights with sensor estimates in the feedback loop controlled the altitude of the robot during hovering flight with altitude error of less than 1.5 cm, or within a half body length of the robot. These experiments also demonstrated that controlled hovering with sensor estimates in the feedback loop had twice the RMS error of the flights with Vicon feedback. To investigate this error, we verified the sensor measurements against the reference height from Vicon (see Fig. 9). The sensor performed as expected across all flights, with less than 5% of the measurements deviating from the linear trend. Outliers where the sensor measurement was lower than the expected value can be explained by an occluded view of the ground from the tether, and areas where the sensor measurement were higher are due to the attitude of the vehicle moving away from the vertical axis. Practically, the robot does not tilt away from the vertical axis more than 15°. At an altitude of approximately 10 cm, this second-order effect will cause deviations in sensor measurement of 4 mm larger than the true value. In the future, accurate attitude estimation may not be available and reliance on these estimates for correction may be impractical. However, given the limited number of outliers, this effect is negligible and the source of error is not sensor measurement.

Previous studies [11] have shown that the power tether can produce large torques on the robot during flight, making the vehicle more difficult to control during hover. The addition of a second power tether for sensor communication only increases this effect. Additional tuning will be necessary to compensate for these effects.

5 Conclusions

Altitude sensing is a necessary component for vision-based navigation. In this work, we determined a relationship between sensor latency and settling time for the closed-loop altitude dynamics of this vehicle, elucidating the feedback requirements for various altitude maneuvers. With this information, we selected a sensor that met the mass, power, and latency constraints of the vehicle to perform altitude estimation in free flight. The closed loop flights with sensor estimates in the feedback loop are the first demonstrations of controlled altitude with onboard estimation for an insect-scale robot in free flight. These flights, coupled with the results from the calibration experiments, demonstrate that the sensor is able to accurately measure altitude, but further work is needed in tuning the controller gains to lower the average error about the desired setpoint.

The low mass of the sensor allows for additional payload, which can be used for combinations of sensors. Future work into sensory fusion, including the integration of an IMU to estimate attitude (as demonstrated in [10]) will enable the RoboBee to perform short hovering flights with only onboard sensory information. Additionally, autonomous visual navigation can be achieved with the integration of an onboard optic flow sensor in combination with an attitude estimate from the IMU and an absolute distance measure from the proximity sensor. These experiments will be the first demonstrations of sensor autonomy on an at-scale robotic insect. This work has provided an important step towards this goal.

This work can also be applied to other MAVs, especially those with stringent payload, power or computational requirements. Given the short range of this sensor, this sensor is impractical for altitude control over a large variation in height. However, a ToF sensor could provide a precise measure of proximity and be combined with a pressure sensor to provide a coarse altitude estimate. We can envision this sensor being used for close range object detection or obstacle avoidance.

Acknowledgements The authors would like to thank Neel Doshi for his help in developing the model for the closed loop dynamics. This material is based upon work supported by the National Science Foundation Graduate Research Fellowship under Grant No. (DGE1144152), and the Wyss Institute for Biologically Inspired Engineering. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation.

References

1. Abzug, M.J., Larrabee, E.E.: Airplane Stability and Control: A History of Technologies that Made Aviation Possible, 2nd edn. Cambridge University Press, Cambridge (2002)
2. Astrom, K.J., Murray, R.M.: Feedback Systems: An Introduction for Scientists and Engineers. Princeton University Press, Princeton (2008)
3. Baek, S.S., Garcia Bermudez, F.L., Fearing, R.S.: Flight control for target seeking by 13 gram ornithopter. In: 2011 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp. 2674–2681. IEEE (2011)
4. Beyeler, A., Zufferey, J.-C., Floreano, D.: Vision-based control of near-obstacle flight. Auton. Robot. **27**, 201–219 (2009). doi:10.1007/s10514-009-9139-6
5. Chirarattananon, P.: Flight control of a millimeter-scale flapping-wing robot (2014)
6. Chirarattananon, P., Ma, K.Y., Wood, R.J.: Adaptive control of a millimeter-scale flapping-wing robot. Bioinspir. Biomim. **9**(2), 025004 (2014)
7. De Wagter, C., Tijmons, S., Remes, B.D.W., de Croon, G.C.H.E: Autonomous flight of a 20-gram flapping wing MAV with a 4-gram onboard stereo vision system. In: 2014 IEEE International Conference on Robotics and Automation (ICRA), pp. 4982–4987. IEEE (2014)
8. Dudley, R.: The Biomechanics of Insect Flight: Form, Function, Evolution. Princeton University Press, Princeton (2002)
9. Duhamel, P.-E.J., Perez-Arancibia, N.O., Barrows, G.L., Wood, R.J.: Biologically inspired optical-flow sensing for altitude control of flapping-wing microrobots. IEEE/ASME Trans. Mech. **18**(2), 556–568 (2013)
10. Fuller, S.B., Farrell Helbling, E., Chirarattananon, P., Wood, R.J.: Using a MEMS gyroscope to stabilize the attitude of a fly-sized hovering robot. In: IMAV 2014: International Micro Air Vehicle Conference and Competition 2014, Delft, The Netherlands, 12–15 August. Delft University of Technology (2014)
11. Fuller, S.B., Karpelson, M., Censi, A., Ma, K.Y., Wood, R.J.: Controlling free flight of a robotic fly using an onboard vision sensor inspired by insect ocelli. J. R. Soc. Interface (2014, in press)
12. Farrell Helbling, E., Fuller, S.B., Wood, R.J.: Pitch and yaw control of a robotic insect using an onboard magnetometer. In: 2014 IEEE International Conference on Robotics and Automation (ICRA), pp. 5516–5522. IEEE (2014)
13. Humbert, J.S.: Bio-inspired visuomotor convergence in navigation and flight control systems. Ph.D. Thesis, California Institute of Technology (2005)
14. Intersil. Digital proximity sensor with interrupt function. Data Sheet (2009)
15. Kumar, V., Michael, N.: Opportunities and challenges with autonomous micro aerial vehicles. Int. J. Robot. Res. **31**(11), 1279–1291 (2012)
16. Ma, K.Y., Chirarattananon, P., Fuller, S.B., Wood, R.: Controlled flight of a biologically inspired, insect-scale robot. Science **340**, 603 (2013)
17. Ma, K.Y., Felton, S.M., Wood, R.J.: Design, fabrication, and modeling of the split actuator microrobotic bee. In: Proceedings of the IEEE International Robotics and Automation Conference (2012)
18. Measurement Specialties. Ms5607-02ba03. barometric pressure sensor, with stainless steel cap. Data Sheet (2012)
19. Murata. Ma40h1s-r. Product Specification of Ultrasonic Transducer (2014)
20. Srinivasan, M.V.: Honeybees as a model for the study of visually guided flight, navigation, and biologically inspired robotics. Physiol. Rev. **91**(2), 413–460 (2011)
21. ST Microelectronics. Vl6180x. proximity and ambient light sensing (als) module. Data Sheet (2014)

Tensile Web Construction and Perching with Nano Aerial Vehicles

Adam Braithwaite, Talib Alhinai, Maximilian Haas-Heger, Edward McFarlane and Mirko Kovač

1 Introduction

Multi-rotor micro aerial vehicles (MAVs) are a subject of intense interest within both the robotics community and society at large, with industrial interest also expanding rapidly. These vehicles have been proposed for a myriad of applications, from airborne surveillance and mapping to large-scale construction projects. The fundamental nature of rotary aircraft lends them the ability to maintain a relatively simple hover and excel at low-velocity, high accuracy manoeuvring. However, such vehicles also exhibit significant drawbacks; they have relatively poor flight endurance, and are highly susceptible to aerodynamic effects and interference from the environment when in close proximity to external surfaces, and thus struggle to maintain static flight in closely confined spaces. In this publication, we address and implement a method to mitigate flight time limitations for rotary MAVs by constructing a structure within the immediate environment and subsequently perching upon the same structure (Fig. 1).

As a means prolonging the endurance of rotary-craft MAVs, where typical flight times are in the order of 10–20 min [1], several research groups have investigated perching as a viable power management solution. Some of the approaches proposed to perch to vertical surfaces and ceilings include using magnetic adhesion [2], micro spines [3–5], electrical adhesion [6, 7], adhesive pads [8, 9], and hot-melt adhesives [10].

A separate set of works has been presented on construction systems utilising aerial robots, usually making use of pre-fabricated components and connectors.

Adam Braithwaite and Talib Alhinai contributed equally to this work.

A. Braithwaite · T. Alhinai (✉) · M. Haas-Heger · E. McFarlane · M. Kovač (✉)
Aerial Robotics Laboratory Department of Aeronautics,
Imperial College London, London, UK
e-mail: talib.al-hinai13@imperial.ac.uk

M. Kovač
e-mail: m.kovac@imperial.ac.uk

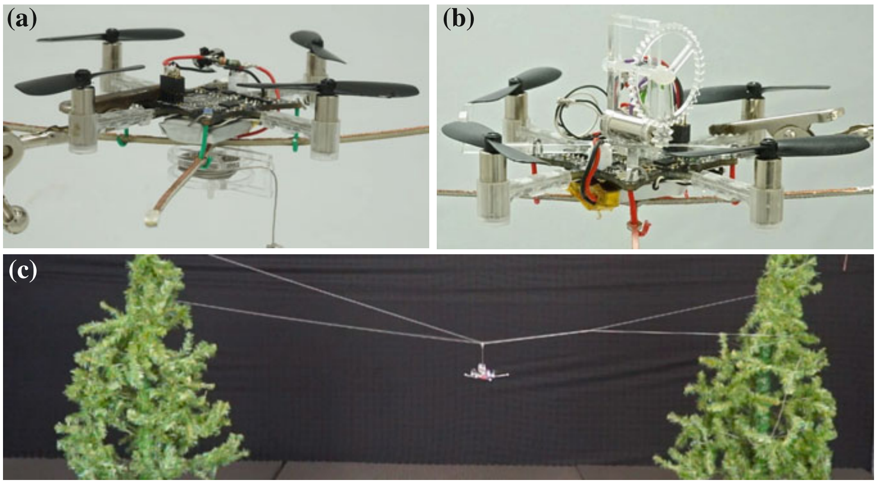


Fig. 1 **a** and **b** Nano quadrotors with a mass of 26 g each carrying construction and perching payloads respectively. The constructor payload consists of a spool of string intended for creation and utilization of a tensile web structure. The percher payload consists of a hook at the end of a connecting string intended to latch on to an overhanging support element. **c** Physical realisation of our web structure, created by NAVs, with a quadrotor perched for an extended duration

For example, [11] demonstrates the assembly of a beam-based structure by flying robots, relying on magnetic connectors to join the various construction components. Another project describes a method by which a mid-sized quadrotor MAV assembles a rope structure between two previously placed rigid, parallel elements [12]. Various other approaches to aerial robotic construction with soft components have also been demonstrated [13, 14]; we recently presented a flying robot equipped with a mechanism to deposit polyurethane foam to create or connect structures in flight [15].

In this paper, we present construction and perching mechanisms to enable multi-rotor MAVs across a wide range of scales to maintain and control their relative altitude whilst perched upon an external structure, requiring no active attitude control and thus the use of only a single motor. To maximize the range of applicability of such a scheme in tightly constrained environments, we decide to focus on sub-miniature multi-rotor MAVs or nano aerial vehicles (NAVs), hereby defining an NAV as a robot with a total mass of no more than 30 g and a span of less than 10 cm. Employment of such a system is envisaged in complex environments in scenarios requiring long-term position maintenance, such as monitoring inaccessible areas of remote rainforests for environmental variations or structural health monitoring of bridge structures. Similarly, such a system may be employed to inspect and monitor for damage around industrial sites otherwise considered difficult to access, such as around and within pipework intrinsic to petrochemical refinery installations. To facilitate such monitoring, additional sensors (light, temperature, or a video camera) may be mounted to these NAVs and activated when successfully perched in the required area.

Although our contribution is conceptually similar to [12]; it differs in three distinct ways. Firstly, our approach to construction is collective. We demonstrate how multiple NAVs can cooperate to complete certain construction tasks. Second, we employ time-stamped trajectories that adapt with respect to the position of physical supports in the environment. Thirdly, we demonstrate a string-based perching mechanism for low-power altitude control. In particular, our approach distinguishes itself in terms of leveraging aerial construction as means of allowing a robot to extend control over an environment with an example shown of an NAV constructing a structure and subsequently using the said structure with the purpose of perching for long-duration monitoring.

2 Equipment and Testing Methodology

The flight arena in the Aerial Robotics Laboratory at Imperial College London is a protected area of 5.5 m by 10.5 m, with a maximum effective flight altitude of 6.2 m. The arena is equipped with 16 Vicon T40-S cameras, each with a sensor resolution of 4 megapixels and system update rate of 400 Hz. The system is capable of tracking retroreflective markers or near-IR LEDs to sub-millimetre accuracy throughout the available flight space, providing high accuracy ground-truth localization for as many individual objects as required.

In addition to this equipment, we also simulate the fixed anchor points we may expect to encounter in a natural environment, such as tree branches or a forest canopy. To create a representative parallel we add two artificial coniferous trees to the arena, each mounted on a fixed wooden base to provide tree-like surfaces 1–2 m above the arena floor.

To maximise the range of applicability of this construction scheme, we require aerial robots of a sufficiently small size to enable access to all but the most constricted areas. Many research-grade MAVs have overall spans of 0.5 m or above [16], but we desire a dramatically smaller platform on which to build. In addition to increasing the range of accessible environments, reducing the scale of the robots also tends to reduce the cost and thus allow the deployment of larger swarms with similar capital expenditure, increasing the robustness and failure tolerance of such a scheme [17].

Following a comprehensive review of the state of the art and market availability, we selected a commercially available system appearing to suit the needs of this task. The Crazyflie nano-quadrotor [18] has an overall span of 9 cm, a 32 bit ARM Cortex M3 microprocessor for flight control, a high-quality inertial measurement unit (IMU) consisting of a 3-axis accelerometer, gyroscope and magnetometer, and a barometer to together provide 10 degree of freedom state measurement. As with virtually any method of pressure measurement operating in non-static flow across small altitude fluctuations, accuracy is relatively poor and thus altitude data is derived entirely from the Vicon measurements. All other sensors are used to provide state estimations of each NAV and thus provide closed-loop attitude control. Each NAV weighs 18.8 g with a 170 mAh lithium polymer (LiPo) battery, providing around 7 min of effective hover time when no additional payload is added.

3 Web Construction

Two distinct payloads are proposed for the NAVs involved in this construction. The first should enable a method of thread attachment to the external environment, a mechanism by which the thread may be deployed simply by the motion of the quadrotor (but retained whilst hovering) and a method of completing the structure and detaching to return to base. The second payload is attached to one or more quadrotors aiming to perch on the structure created, and must therefore provide a method of suspending the NAV in a state from which flight can be consistently recovered by the control scheme employed. A mechanism to control the height of the quadrotor below the structure is also required to extend the capability of such suspended NAVs, but this must exhibit the minimum possible energy consumption to demonstrate significant advantages over simple hovering.

We use the term *web* to describe the tensile structures we aim to create, consisting of multiple interlinked non-rigid elements. When assembling and perching on a web, we are especially interested in the three main properties: (**1**) Accuracy of the NAV trajectories and hence accuracy of placement and geometric characteristics of the web produced. (**2**) Structural properties of the web, including maximum loading in a vertical plane to characterise the ability of the structure to support one or more perched NAVs below. (**3**) The ability of the perching NAVs to autonomously approach, mount, dismount and leave the web structure without causing material detriment to the long-term integrity of the web.

We will next consider how to best implement and optimize two modular payloads enabling NAVs to specialize in different tasks. We will use the term *constructor* and *percher* to refer to the packages designed to enable the NAVs to construct the web structure and dock to a component tensile element of the structure when completed, respectively.

3.1 Construction

To enable assembly of the web, we define our target process to be the robust deployment of multiple discrete tensile elements between pre-existing structures. The constructor payloads aim to enable deployment of the web material through only applied tensile force carried by the material itself; that is, the deployment mechanisms on the modules themselves are entirely passive. We therefore require the initial force for deployment to be greater then the force required to overcome the static friction

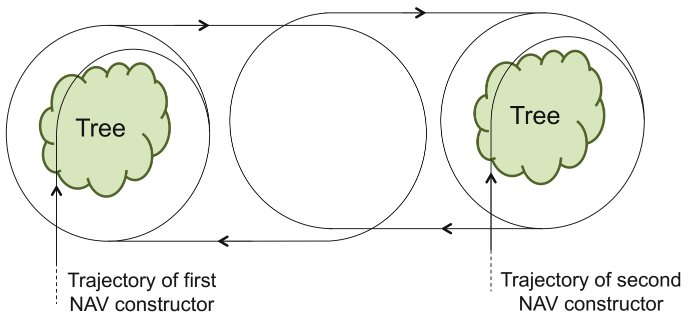

Tree
Tree
Trajectory of first NAV constructor
Trajectory of second NAV constructor

Fig. 2 Web construction about two arbitrary tree structures. The flexible structural elements are first secured to the anchor point by means of a lightweight hook, before the NAV winds the thread around the tree for as many complete rotations required to support the desired loading. The NAVs then complete a linking node in the centre before returning to secure the web

between the spool and the supporting central rod and hence begin to unspool the thread from the construction module, approximately 23 mN.

To model unstructured anchor points to simulate those present in a natural environment, we employ two artificial trees mounted on supports to reach a peak height of around 2 m (Fig. 2). The foliage of these trees is arranged in a relatively unstructured pattern (with no convenient protruding sections for attachment) to best emulate coniferous woodland in the real world. To enable reliable connection with these structures, we choose to design a small hook to embed itself within the foliage of the tree and provide initial resistance to small forces. This connection may then be further reinforced via trajectory winding methods discussed later in this paper. The precise configuration of attachment mechanism employed is heavily influenced by the current experimental set-up (that is, the use of these particular tree analogues), but the method may be optimized or altered to suit differing environments, such as small magnets for metallic objects or larger hooks for less dense foliage - we therefore present all methods and mechanisms other than the particular hook design without loss of generality. Our hooks feature 4 prongs in a radial configuration with a total diameter of 14 mm with a tooth inclination of 50°. Web material choice was based on weight, strength and roughness/static friction. We require sufficient roughness to sustain a sufficient reaction force by static friction against the tree foliage to oppose that exerted downwards when other NAVs perch upon the structure. Silk thread, 0.35 mm diameter, suited this criteria with 8 m of thread weighing under 1 g and providing a high enough coefficient of friction to resist an applied load of 85 g in the web under testing, sufficient to support three perched NAVs.

For stability and ease of control, we desire the centre of gravity to be coincident with the centre of force applied to the NAV - for these symmetric quadrotors in unrestrained flight, this means directly below the centre of the body. The battery is positioned below the centre of the NAV with the construction pack extended directly below this. It carries a spool capable of holding 8 m of silk threading with a separately rotating arm assisting in deployment and controlling the release point to minimize the magnitude of out-of-plane forces on the thread ($\mathbf{F}_n$, $\mathbf{F}_b$ in the normal and binormal directions respectively). Assuming these forces are small compared to the in-plane component, $\mathbf{F}_t$, the torque, τ, or moment induced on the quadrotor by the tension force in the thread is linearly related to the distance, $\mathbf{r}$, to the quadrotor centre of mass.

Although the pitch and roll moments exerted on the NAV should be relatively time-invariant if a constant force is applied to the thread, the torque about the vertical body axis will vary as string is removed from the reel and thus the effective separation between the detachment point and the centre of mass changes. As a result, we specify heading direction as unconstrained, allowing any orientation in flight and therefore meaning the controller is relatively uninfluenced by this variable load. The empty mass of the constructor payload is 0.89 g, with a maximum capacity of 8 m of silk string giving a loaded mass of approximately 2 g.

3.2 Perching

The percher payload allows NAVs to dock to the web and maintain active vertical control by spooling and despooling the connected thread. The docking procedure is accomplished by an NAV following a trajectory perpendicular to the web approximately 5 cm above the intended point of attachment, with the connection thread suspended a short distance below. The NAV moves across the web from one side, and once crossing slowly decreases the motor thrust whilst maintaining slow forward flight to avoid any interference between the thread and propellers. A hook at the end of the connecting thread then catches the web and creates a robust attachment between the web structure and the perched NAV (Fig. 3). To avoid the entanglement of the thread with the propellers, we maintain a $\pi/4$ yaw offset from the direction of the thread on which to be perched - provided the tension is maintained, the ensures that the thread must pass between two arms and the $\sim$2 cm gap between the propellers.

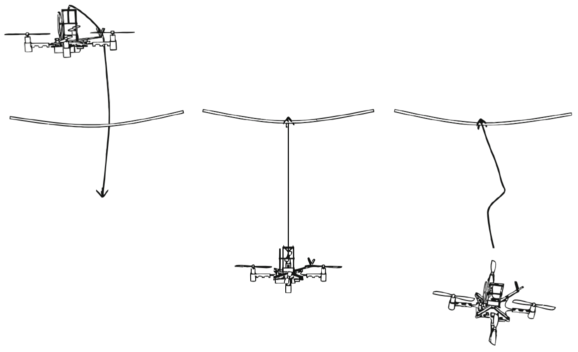


Fig. 3 NAV with perching payload attaching to tensile structure, suspending below to enable passive maintenance of position, and displaying one method of dismounting from the structure and retaking active flight, particularly suited to constrained environments with limited horizontal space

To resume flight, we may either de-spool the remaining thread from the percher module thus detaching from the end of the thread, or effectively perform the docking procedure in reverse. This second method is functionally preferable as the system is then capable of re-perching on the web in future, but the mechanism chosen in any particular scenario is dependent upon environmental constraints. The first method to dismount requires sufficient vertical space to fully extend the thread, but negligible horizontal space, whilst the second method requires less vertical space but a greater horizontal travel distance perpendicular to the web to ensure the connecting thread is continually taut.

The percher module incorporates a brushed DC motor to enable active altitude control whilst suspended from the thread - given the excellent power/weight ratio of the Crazyflie motors, we elect to use one more of these for this purpose. The motor torque required to sustain ascent of the NAV whilst perched increases linearly with the effective diameter of the thread spool. Considering a maximum permitted NAV mass of 25 g (set by the flight payload limit), we prescribe a maximum spool diameter of 6 mm, allowing approximately 2 m of thread. We require the motor exert sufficient force to exceed both the static friction and the force required to lift the NAV, approximately 0.3 N, and so employ a gear mechanism to increase the effective torque by a factor of 5. The rotational speed is controlled by pulse width modulation of the motor, at 100 Hz, varying speed by altering the duty cycle.

Damping of vertical motion along the connecting thread is achieved by by a combination of static friction and active, motor-enabled braking. The static friction of the system is sufficient to maintain the altitude of the NAV whilst stationary, allowing station keeping with minimal system current draw used during standby (< 10 mA). For comparison, active hovering requires more than 2 A from the 4 flight motors, at minimum payload weights. Dynamic manoeuvres whilst perched require active braking. Active braking is achieved by our motor controller, applying up to 100 mA to rapidly damp motion when required. Power and control to the modules is provided through an expansion header interface attached to the Crazyflie NAV.

4 Trajectory Planning and Control

4.1 Dynamic Model

Using a Cartesian co-ordinate system, we define the quadrotor body frame as B and the world frame as W. We choose to dissociate the yaw rotation (ψ) from the local pitch and roll angles (ϕ and θ respectively) and thus decompose our rotation matrix in SO(3) into two separate components, ${}^{W}R_{S}$ and ${}^{S}R_{B}$. The frame S is an intermediate state, accounting for the yaw rotation of B relative to W but not pitch or roll.

$$ {}^{W}R_S = \begin{bmatrix} cos\psi & -sin\psi & 0 \\ sin\psi & cos\psi & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{1} $$

$$ {}^{S}R_B = \begin{bmatrix} cos\theta & sin\phi sin\theta & cos\phi sin\theta \\ 0 & cos\phi & -sin\phi \\ -sin\theta & sin\phi cos\theta & cos\phi cos\theta \end{bmatrix} \tag{2} $$

$$ {}^{W}R_B = {}^{W}R_S \, {}^{S}R_B \tag{3} $$

We denote the angular velocity about the body axes $[x_B, y_B, z_B]^T$ as $[p_B, q_B, r_B]^T$. The rotational velocity relative to the world frame, henceforth denoted $\mathbf{v}_{BW}$ for brevity, may now be expressed as a function of the body angular velocity and these two rotation matrices. Note that the angular velocity in the body frame is considered analogous to the derivatives of the body frame roll, pitch and yaw angles prior to transformation.

$$ \mathbf{v}_{BW} = {}^{W}R_B \begin{bmatrix} p_B \\ q_B \\ r_B \end{bmatrix} \tag{4} $$

The angular acceleration of the body relative to the world frame is then computed via Euler's rigid body equations, where I is the body inertia matrix aligned with body co-ordinates and centred at the body centre of mass and L is the length of the moment arm, in practise the distance between each motor and the body centre of mass. The force F_i and moment M_i are modelled as functions of the square of motor angular velocity (ω^2).

$$ I\dot{\mathbf{v}}_{BW} = \begin{bmatrix} L(F_2 - F_4) \\ L(F_3 - F_1) \\ M_1 - M_2 + M_3 - M_4 \end{bmatrix} - \begin{bmatrix} p_B \\ q_B \\ r_B \end{bmatrix} \times I \begin{bmatrix} p_B \\ q_B \\ r_B \end{bmatrix} \tag{5} $$

Now considering the linear dynamics, we denote the position vector of the body centre of mass relative to the world frame as $\mathbf{r}$. From a free body approach, the forces acting on the system are gravity, acting vertically downwards in the world frame, and the lift contribution from each propeller acting vertically upwards in the body frame. Applying an appropriate sequence of rotations as discussed above and applying Newton's equations of motion gives a model of the linear dynamics of the quadrotor as a function of ${}^{W}R_B$, the body mass m and the force contribution from each propeller, F_i.

$$ m\ddot{\mathbf{r}} = \begin{bmatrix} 0 \\ 0 \\ -mg \end{bmatrix} + {}^{W}R_B \begin{bmatrix} 0 \\ 0 \\ \Sigma F_i \end{bmatrix} \tag{6} $$

Whilst the quadrotor is deploying the thread as it is held in tension against the previous node, we must consider the effect of this tensile force upon the flight dynamics of the NAV. Relative to the rotational centre of the quadrotor, this will result in both a linear force in an arbitrary direction and potential moments in all rotational degrees of freedom. These moments are determined by the angles subtended by the string relative to the quadrotor in three-dimensional space, and the position of the thread guide hole in the arm of the construction pack with respect to the centre of the vehicle. We denote the angle of the incoming thread as ϕ_t, θ_t and ψ_t in the body roll, pitch and yaw axes respectively, assuming a linear link between the current position of the quadrotor and the centre of the previous node - for linking nodes, this is the target intercept of both thread elements, and for attachment to each tree we assume the thread to extend from the centre point of the circular winding trajectory used to create the attachment. Both assumptions here are materially valid provided we keep the thread as taut as possible throughout the process, and the winding radius about each tree is relatively small; in practise, we find that the thread will leave the tree from within around 5 cm of the target centre point. By denoting η_a as the angle of the arm relative to the positive longitudinal body axis of the NAV, $|F|$ as the magnitude of the string tension and L_P as the construction pack arm length, the contribution to pitching moments induced by the tension in the thread in body axes (M_ϕ, M_θ, M_ψ) can be expressed as:

$$\mathbf{M} = \begin{pmatrix} M_\phi \\ M_\theta \\ M_\psi \end{pmatrix} = |F| L_P R \begin{pmatrix} sin\eta_a \\ cos\eta_a \\ 1 \end{pmatrix} \tag{7}$$

$$R = \begin{bmatrix} c\phi_t c\theta_t - s\phi_t s\psi_t s\theta_t & -c\phi_t s\psi_t & c\psi_t s\theta_t + c\theta_t s\phi_t s\psi_t \\ c\theta_t s\psi_t + c\psi_t s\phi_t s\theta_t & c\phi_t c\psi_t & s\psi_t s\theta_t - c\psi_t c\theta_t s\phi_t \\ -c\phi_t s\theta_t & s\phi_t & c\phi_t c\theta_t \end{bmatrix} \tag{8}$$

where c and s denote cos and sin, respectively. Although this offers a reasonable estimation, the precise temporal magnitudes and directions of these loads in reality are difficult to model accurately, as they depend on numerous factors not considered here including roughness of the particular section of thread, and thus added forces from friction, and any movement of the element attachment to the anchor point at the previous node, therefore changing the computed relative angles of thread and quadrotor. In practise, we assume that $|F|$ is reasonably constant throughout the thread extrusion process, measured at 23 mN ($\pm$2 mN) in static testing, and therefore take this as a constant value throughout all web construction segments of the flight. We derive the incident thread angles by assuming a linear link between the centre of the target attachment point and the centre of the NAV, and assume an attachment is made between the thread and the anchor point when the NAV passes within 0.2 m of the horizontal centre of the target.

4.2 Attitude Control

The desired orientation matrix, ${}^{W}R_B$ is generated from a row of direction vectors in the body frame, the values of each being derived from the orientation of the desired acceleration vector ($\ddot{\mathbf{r}}_t$) provided by a higher-level position controller.

$$ {}^{W}R_B = [\mathbf{x}_B, \mathbf{y}_B, \mathbf{z}_B] \tag{9} $$

We first observe that the thrust force generated by the quadrotor always acts vertically upwards in the body frame, and so we must stipulate that the third component of the body axes z_B must be aligned with the desired acceleration vector. This therefore generates the first of our 4 required control inputs - the desired force vector - $u_1 = m\ddot{\mathbf{r}}^t \cdot \mathbf{z}_B - \mathbf{F}_t$ where $\mathbf{z}_B$ is a unit vector in the z_B direction and $\mathbf{F}_t$ the force vector generated by the tension in the string. The error in the $\ddot{\mathbf{r}}^t$ direction is therefore purely in z_B, and we can hence define the unit vector of the error as $\mathbf{e}_3 = [0, 0, 1]^T$ so that the desired rotation matrix ${}^{W}R^t_B$ must align this vector with the target $\mathbf{z}^t_B$.

$$ {}^{W}R^t_B \mathbf{e}_3 = \mathbf{z}^t_B \tag{10} $$

Given the specified yaw angle from the position control, ψ_t, we may easily compute the desired orientation of the $\mathbf{x}_S$ and $\mathbf{y}_S$ unit vectors in the frame S (as discussed earlier, this frame incorporates yaw components but not pitch and roll) via a standard 2D rotation.

$$ \begin{bmatrix} \mathbf{x}^t_S \\ \mathbf{y}^t_S \end{bmatrix} = \begin{bmatrix} cos\psi_t & sin\psi_t \\ -sin\psi_t & cos\psi_t \end{bmatrix} \begin{bmatrix} \mathbf{x}^t_W \\ \mathbf{y}^t_W \end{bmatrix} \tag{11} $$

The desired orientation of y_B, $\mathbf{y}_{B,t}$ is now given by the unit vector in the normal direction to $\mathbf{z}^t_B$ and $\mathbf{x}^t_C$, and finally the desired orientation of x_B computed as a function of the other two vectors to complete the Cartesian set.

$$ \mathbf{y}^t_B = \frac{\mathbf{z}^t_B \times \mathbf{x}^t_C}{||\mathbf{z}^t_B \times \mathbf{x}^t_C||} \tag{12} $$

$$ \mathbf{x}^t_B = \mathbf{y}^t_B \times \mathbf{z}^t_B \tag{13} $$

The orientation error is defined as $\mathbf{e}_r$, and given as a function of the target and current orientation matrices. We also define a corresponding error in the body angular velocity $\mathbf{e}_v$ to permit the implementation of a PD-style feedback controller.

$$ \mathbf{e}_r = \frac{1}{2}({}^{W}R^{tT}_B \, {}^{W}R_B - {}^{W}R^T_B \, {}^{W}R^t_B)^{\vee} \tag{14} $$

$$ \mathbf{e}_v = {}^{B}R_W(\mathbf{v}^t_{BW} - \mathbf{v}_{BW}) \tag{15} $$

Note that $^\vee$ represents an inverted hat-map, or vee-map, which transforms the system from SO(3) to a vector $\mathbf{e}_r \in \mathbb{R}^3$. We then derive the desired moments about each body frame axis (our remaining three control inputs) using PD feedback on these errors, for $i = 2 : 4$. At this stage, our estimated induced moments from the dispensing thread are subtracted from the control inputs to compensate for the external forces acting on the NAV.

$$u_i = k_p \mathbf{e}_{r,i} + k_d \mathbf{e}_{v,i} - \mathbf{M}_i \tag{16}$$

An initial offset to achieve theoretical hover must be added to the speed target for each motor, and is easily calculated for a free-flying NAV by a simple force balance: equating the quadrotor weight mg projected along the z_B axis, and the total force generated by all rotors rotating at an equal and constant speed to balance the expression. This offset is denoted as $u_{1,0}$ below:

$$u_{1,0} = \sqrt{\frac{mg\mathbf{z}_W \cdot \mathbf{z}_B}{4k_F}} \tag{17}$$

$$\begin{bmatrix} \omega_1^t \\ \omega_2^t \\ \omega_3^t \\ \omega_4^t \end{bmatrix} = \begin{bmatrix} k_1 & 0 & -k_1 L & k_2 \\ k_1 & k_1 L & 0 & -k_2 \\ k_1 & 0 & k_1 L & k_2 \\ k_1 & -k_1 L & 0 & -k_2 \end{bmatrix} \begin{bmatrix} u_{1,0} + u_1 \\ u_2 \\ u_3 \\ u_4 \end{bmatrix} \tag{18}$$

A transformation matrix dependent upon the geometry of the quadrotor arms - this case assumes four, equally spaced arms numbered anti-clockwise with the first rotor on the positive x_B axis - then maps the control inputs appropriately. The constants k_1 and k_2 match the control outputs to physical forces and moments exerted by the particular motor/propeller combination on these NAVs, linearised about the hover point.

4.3 Position Control

When constructing tensile structures using ropes or wires we distinguish between two types of node. Nodes placed on pre-existing environmental elements (such as the trees employed here) act as supports to the structure and hence are referred to as supporting nodes. Nodes created by two or more quadrotors interlinking their respective web elements as to connect two tensile structures are referred to as linking nodes. The methodology used to create a node depends on which type of node is to be created; we devise trajectory elements for each.

The purpose of a supporting node is to serve as a rigid, load-bearing anchor to the tensile structure. It must therefore sustain tensile forces arising due to both the weight of the structure itself and any payload attached. To create a node capable of

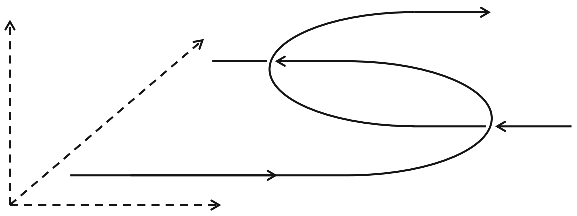


Fig. 4 Linking node created by intersection of trajectories of two NAVs each carrying a trailing structural element. The tensile components intersect and are subsequently pulled tight and anchored by both NAVs to create a complete tensile structure capable of supporting loads

withstanding these tensile forces, we consider the friction effect between the web component and the supporting body. By creating multiple turns around an object, the frictional force may be increased to a sufficient level to support the desired structure. To model the required number of turns for any given element, we consider the expected forces in each element of the web and then refer to the Capstan equation [19]. For tension applied to the two ends of a string, T_1 and T_2, static coefficient of friction μ, and total angle swept by all turns of the string θ, the equation dictates that the string will remain static if the following condition is satisfied:

$$T_2 < T_1 e^{\mu|\theta|} \tag{19}$$

The first manoeuvre required to create such a node is a close pass of the tree to attach the construction hook and provide an initial attachment between the tree and the web. Multiple encirclements of the anchor point then follow to deposit the required length of string for the tensile forces expected, as determined by Eq. 19. The angle θ is directly related to the number of encirclements and hence the tensile strength of the support increases exponentially with the number of revolutions performed.

A linking node (Fig. 4) connects two structural elements, each attached to their respective supporting nodes. To achieve this, two NAVs approach each other on offset trajectories until they reach the point of minimum separation between them. They then fly opposing semicircles in the horizontal plane so as to switch positions and create a link between their respective web elements. The altitude of both quadrotors is linearly increased by a total of 0.5 m throughout this manoeuvre to avoid the collision of any vehicle with the string deposited by the other. Both NAVs return to their respective original positions horizontally aligned with the supporting nodes will to pull the thread components taut. The tension inherent in the resulting linked structure will equal the tension required to unroll additional thread from the construction module, which is then formed into a further supporting node to complete this element of the structure. Linking nodes between more than two elements may be created by employing more than two quadrotors in the manoeuvre described - one for each element.

We employ a PID-based feedback controller to act on all linear degrees of freedom. By tuning two sets of gains, one exhibiting stiff behaviour for optimal trajectory

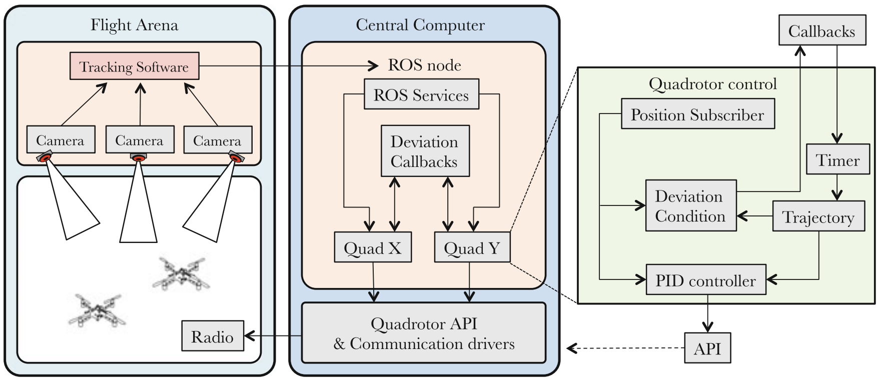


Fig. 5 System architecture for our NAV construction team. We utilise the ROS package to interface with the Vicon motion capture system and all NAVs through a single radio dongle, running individual trajectory controllers, timers and error checkers for each NAV before passing output commands through the Crazyflie API

accuracy, and one softer system permitting rejection of disturbances of up to 4 body lengths, we optimise each section of the trajectory individually. The stiff controller is employed in all free-flight and attachment manoeuvres, as well as at the apex of linking nodes when tensile elements are temporarily allowed to slacken; this ensures accurate and robust attachment when creating nodes, where significant deviations from the desired path would cause the manoeuvre to fail. The softer controller is used for all flight involving the active maintenance of tension in structural elements, to permit the NAV to recover from any significant disturbance generated by the structural forces exerted upon it, whilst maintaining flight stability. This scheme works in tandem with the deviation threshold limit described below to complete the entire flight control package, as outlined in Fig. 5.

4.4 *Trajectory Definition*

We devise trajectories to create a bridging structure between two trees by placing a supporting node on each tree and a linking node equidistant between them. Thus, two NAVs are required to each place a supporting node, co-operatively create one linking node and then return to form two further supports and complete the structure. The desired trajectories for each quadrotor are defined parametrically in time by independent three-dimensional functions for each NAV in the team. These trajectory functions are defined in abstract scales and overall horizontal rotations, and thus provided the location of each tree is known, may be scaled to any configuration and separation; limited only to distances permissible by the length of thread carried on each NAV.

$$\mathbf{r}_T = \begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} f_{i,x}(t) \\ f_{i,y}(t) \\ f_{i,z}(t) \end{pmatrix} \tag{20}$$

Collaborative construction between multiple NAVs requires careful synchronisation of the trajectories to achieve reliable creation of linking nodes. Therefore, we synchronise the time in each NAV controller with a common timer. This ensures all quadrotors reach critical points along their trajectories - for example, points of interaction with other vehicles - at the correct time and simultaneously. Furthermore, we implement a system that detects and acts upon any significant deviation an NAV might have from its flight path. This deviation is calculated in every time step of the control scheme, neglecting errors tangential to the trajectory. If, at an instant in time, we define the position of any given quadrotor as $\mathbf{r}$ and the closest point on its trajectory as $\mathbf{r}_T$, then the effective deviation $\mathbf{e}$ is given by

$$\mathbf{e} = ((\mathbf{r}_T - \mathbf{r})\cdot\hat{\mathbf{n}})\hat{\mathbf{n}} + ((\mathbf{r}_T - \mathbf{r})\cdot\hat{\mathbf{b}})\hat{\mathbf{b}} \tag{21}$$

where $\hat{\mathbf{n}}$ and $\hat{\mathbf{b}}$ are unit vectors in the trajectory normal and binormal directions respectively. Two callback functions in the main program thread on the central control PC are connected to two functions within the control threads of each individual NAV. A deviation magnitude above 0.2 m triggers the first function and hence the corresponding callback. The time at which the deviation threshold was breached is passed as an argument. The internal timer in the control thread of each NAV in the team is paused, thus retaining the time at which the deviation threshold was breached; this effectively pauses every trajectory. If the deviation magnitude of a previously off-course quadrotor falls below 0.1 m, it is deemed to have returned to its flight path and the second function is triggered. As an argument, the callback receives the time elapsed since deviation. The timers governing each quadrotor restart with an offset equal to the time elapsed whilst paused; all NAVs therefore resume construction in a synchronised manner and the team continues the build.

5 Results and Discussion

5.1 Trajectory Accuracy

The trajectory, as observed by the Vicon system, of one NAV throughout the construction is shown in Fig. 6, while the construction sequence is shown in Fig. 7. We note that performance and accuracy of the trajectories can vary significantly in the event the thread snags the foliage during anchoring, hence presenting the need for a deviation condition. Deviations from the desired flight path of up to 0.2 m during construction stages are attributed to the intentional softness of the position controller during these phases of flight. As intended, stability is maintained throughout the

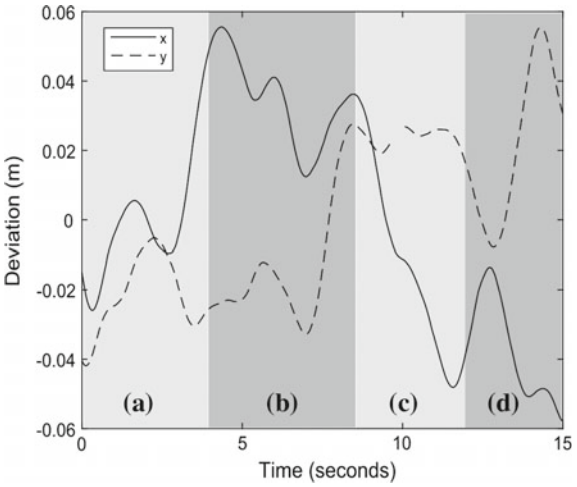


Fig. 6 Trajectory of the x-y positions of one NAV in the process of creating a two-element tensile structure. **a** String is attached to tree branch. We engage the force compensation once our distance threshold is reached, but a delay in hook attachment and thread tensioning likely creates the noticeable deviation at this stage. **b** Supporting node being constructed by the NAV. **c** Free flight away from the anchor point. Our assumption that the force vector is from the centre of the anchor point likely causes the deviation here - where in actuality the force vector is exerted from some point along a branch. **d** Two NAVs form linking node. Good estimation of external forces occurs here, and the node location shifts along as NAVs fly away. Errors at single mission-critical points are less than 0.05 m

construction process despite significant tensile forces in the structural elements and the accuracy in directions of interest is maintained within 0.05 m during all mission-critical manoeuvres, including initial connection of the supporting nodes in all three spacial axes and the trajectory altitude about the apex of the linking node.

5.2 *Structural Properties of the Web*

The web was constructed with two 8 m strands of silk thread that were individually deployed by a pair of constructor NAVs. The complete construction of the structure described here took approximately 120 s, well within the capability of these platforms even allowing for up to 90 s of travel in each direction before building commences. The constructed structure spans 2.1 m, the equivalent of twenty-three NAVs in length. The sequence of the construction strategy is detailed in Fig. 7.

In terms of structural integrity, the web was able to resist an applied load of 85 g under testing, sufficient to support three perched NAVs. We identify the minimum number of loops required around the anchor points as two for the previous load to hold true. However, we note that the uneven surface of the foliage poses a significantly

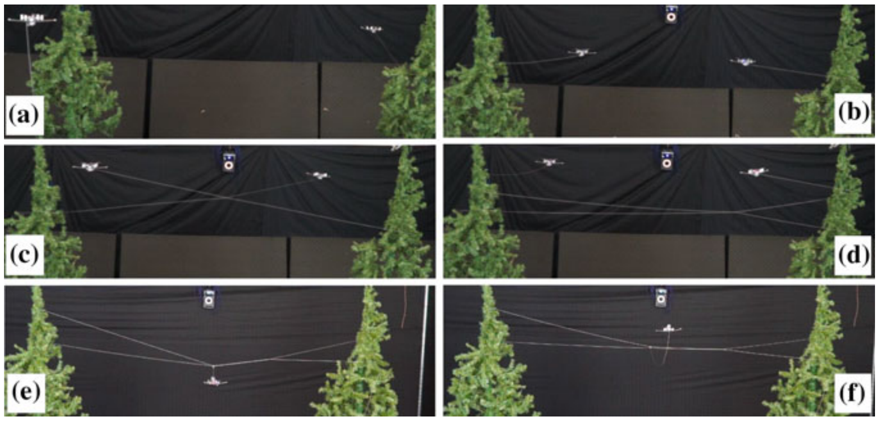


Fig. 7 Two element web construction. Each tensile element is anchored to an environmental feature (**a**), before winding around the anchor point several times (**b**) to create a supporting node. Each element is then pulled towards the centre to create a linking node (**c**), before each drone returns to its original anchor point to secure the web (**d**). The percher drone then approaches and connects to the web, deactivating all rotors and passively maintaining position (**e**) before regaining active control and detaching from the web to return to base (**f**)

reduced contact area to that assumed in the Capstan equation, and thus we add three more windings as a safety margin. These additional loops are a function of the surface properties of the desired anchor point.

5.3 Perching and Energy Consumption

Thrust measurement of a single NAV motor achieved a peak thrust of 73.8 mN during free flight with no significant environmental interactions such as ground or ceiling effect. To assess available hover time, we commanded a single drone to maintain an altitude of 2 m and a fixed horizontal hover position and heading until it was incapable of maintaining this hover due to the draining battery; across 10 trials, we achieved an average of 420 s ($\pm$30 s), with a 170 mAh single-cell 3.7 V lithium-polymer source when no payload was added. When loaded with our construction and perching payloads, the available hover time was reduced to 350 ($\pm$30 s) and 310 s ($\pm$31 s) respectively.

When perched, we measure the current draw of the NAV whilst maintaining a constant altitude at 9.7 mA, rising to 100 mA when actively ascending along the suspending thread. We successfully demonstrate flights of perching NAVs consisting of an 80 s hover to simulate travel, 1 h of perching on a web constructed by two other NAVs (Fig. 7), and a further 80 s hover after dismounting before the battery cell is unable to supply sufficient power to maintain further flight. A similar example is also demonstrated in the video attachment to this paper, albeit with a much shorter time while perched on the web. Assuming identical travel before reaching the target hover position, this represents an overall jump for hover duration from 260 s for an unloaded NAV maintaining active flight to over 3600 s whilst perched on the assembled aerial structure - an increase of nearly 1300%, aptly demonstrating the potential to prolong the effective mission time of inspection or monitoring tasks using the principles we propose.

6 Conclusions

In this paper, we demonstrate the feasibility and range of application of a collaborative aerial construction scheme using sub-10 cm multi-rotor NAVs and multiple element tensile structures. We discuss the methods of securely mounting the web structure to the environment as supporting nodes and assembling multiple components as linking nodes. A mechanism by which an NAV may perch on the assembled structure is then discussed, implemented and assessed, leading to a dramatic increase in overall flight endurance compared to active hovering.

We plan to extend the work presented here to encompass more complex structures, with more anchor points and greater loading capacity to allow higher numbers of perched NAVs. We also expect to investigate applications for this technology, including the mounting of environmental sensors on perched NAVs and applicability in non-natural environments.

Acknowledgements The authors gratefully acknowledge the Abu Dhabi Investment Authority (ADIA) for financial support of this work. The authors would also like to extend their thanks to the participants, scientific panel and technical staff at the Department of Aeronautics at Imperial College London for their enthusiasm and support during the MEng Group Design Project, where some of the ideas presented in this paper were initially explored.

References

1. Roberts, J., Zufferey, J., Floreano, D.: Energy management for indoor hovering robots. In: IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp. 1242–1247 (2008)
2. Yanagimura, K. et. al.: Hovering of MAV by using magnetic adhesion and winch mechanisms. IEEE Conf. Robot. Autom. 6250–6257 (2014)
3. Lussier-Desbiens, A., Cutkosky, M.: Landing and perching on vertical surfaces with microspines for small unmanned air vehicles. J. Intell. Robot. Syst. **57**(1), 313327 (2010)
4. Kovac, M., Germann, J., Hurzeler, C., Siegwart, R., Floreano, D.: A perching mechanism for micro aerial vehicles. J. Micro-Nano Mechatron. (2010)
5. Mellinger, D., Shomin, M., Kumar, V.: Control of quadrotors for robust perching and landing. In: International Powered Lift Conference (2010)
6. Prahlad, H. et. al.: Electro adhesive robots - wall climbing robots enabled by a novel, robust, and electrically controllable adhesion technology. IEEE International Conference on Robotics and Automation, pp. 3028–3033 (2008)

7. Huan, P. et. al.: Characterization of Electro-adhesives for robotic applications. In: IEEE International Conference on Robotics and Biomimetics, pp. 1867–1872 (2011)
8. Unver, O. et. al.: Geckobot: a gecko inspired climbing robot using elastomer adhesives. In: IEEE International Conference on Robotics and Automation, pp. 2329–2335 (2006)
9. Estrada, M.A., Hawkes, E.W., Christensen, D.L., Cutkosky, M.R.: Perching and crawling: design of a multimodal robot. In: IEEE International Conference on Robotics and Automation (2014)
10. Wang, L., Culha, U., Iida, F.: A dragline-forming mobile robot inspired by spiders. Bioinspir. Biomim. **9**, 016006 (2014)
11. Lindsey, Quentin, Mellinger, Daniel, Kumar, Vijay: Construction of cubic structures with quadrotor teams. Auton. Robots **33**(3), 323–336 (2012)
12. Augugliaro, F., et al.: Building tensile structure with flying machines. In: IEEE/RSJ International Conference on Intelligent Robots and Systems (2013)
13. Willman, J., et al.: Aerial robotic construction: towards a new field of architectural research. Int. J. Architect. Comput. **10**(10), 439–459 (2012)
14. Aerial Robotics Cooperative Assembly System (ARCAS). http://www.arcas-project.eu, April 27th (2015)
15. Hunt, G., Mitzalis, F., Alhinai, T., Hooper, P., Kovac, M.: 3D printing with flying robots. In: International Conference on Robotics and Automation. Hong Kong (2014)
16. Ascending Technologies GmbH. http://www.asctec.de, April 27th (2015)
17. Werfel, J., Petersen, K., Nagpal, R.: Designing collective behaviour in a termite-inspired robot construction team. Science **343**(6172), 754–758 (2014)
18. The Crazyflie Nano-Quadrotor. http://www.bitcraze.se, April 27th (2015)
19. Stuart, I.M.: Capstan equation for strings with rigidity. Br. J. Appl. Phys. **12**(10), 559 (1961)

Analytical SLAM Without Linearization

Feng Tan, Winfried Lohmiller and Jean-Jacques Slotine

1 Introduction

Simultaneous localization and mapping (SLAM) is a key problem in mobile robotics research. The Extended Kalman Filtering SLAM (EKF SLAM) approach is the earliest and perhaps the most influential SLAM algorithm. It linearizes a nonlinear SLAM model so that Kalman Filter can be used to achieve local approximate estimations. However, this linearization process on the originally nonlinear model introduces accumulating errors which causes the algorithm to be inconsistent and divergent [1, 2]. Such inconsistency will be particularly prominent in large-scale estimations, resulting in over-optimistic results. Moreover, the quadratic growth of the size of the covariance matrix with the number of landmarks makes the algorithm inscalable to large datasets.

This paper proposes an innovative approach to the SLAM problem by introducing virtual measurements into the system. Completely free of linearization, this approach yields simpler algorithms and guaranteed convergence rates. The virtual measurements open up the possibility of exploiting LTV Kalman-filter and contraction analysis tools in combination in SLAM problem, and information from landmarks or features can be recursively incorporated.

The proposed algorithm is global and exact, which affords several advantages over the existing ones. First, contraction analysis is used for convergence and consistency analysis of the algorithm. As a result, exponential and global convergence rates are guaranteed. Second, conditioned on the local inertial coordinates of the robot,

F. Tan · W. Lohmiller · J.-J. Slotine (✉)
Massachusetts Institute of Technology, Cambridge, MA, USA
e-mail: jjs@mit.edu

F. Tan
e-mail: fengtan@mit.edu

W. Lohmiller
e-mail: wslohmil@mit.edu

A. Bicchi and W. Burgard (eds.), *Robotics Research*, Springer Proceedings
in Advanced Robotics 2, DOI 10.1007/978-3-319-51532-8_6

the algorithm decouples the covariances on each pairs of landmarks, which shrinks the covariance matrix and allows large database applications. Third, the algorithm is simple and straightforward mathematically; it exploits purely linear kinematics constraints that are intuitive and effective. Finally, the algorithm is fast because it decouples the covariances between landmarks and treat each of them independently. Only computations in small scale will be involved to predict and update the landmarks states. We prove the capability of our algorithm in providing accurate estimations in both 2D and 3D settings by applying the proposed framework to four different combinations of sensor information, ranging from traditional bearing measurements and radial measurements to novel ones such as optical flows and time-to-contact measurements.

In the remainder of this paper, we first provide a brief survey on existing SLAM methods in Sect. 2. Basic tools in contraction theory are provided in Sect. 3. Our algorithm with four application cases utilizing different combinations of sensor information is introduced in Sect. 4. Simulation results are presented in Sect. 5. We conclude and discuss the results in Sect. 6.

2 A Brief Survey of Existing SLAM Results

In this section we provide a brief introduction on the problem of simultaneous localization and mapping (SLAM). We will review the three most popular categories of SLAM methods: the extended Kalman filter SLAM, the particle SLAM and graph-based SLAM, each with strength and weakness analysis. Then we introduce the azimuth model that is used in this paper along with the kinematics models describing the locomotion of a mobile robot and the landmarks.

Simultaneous localization and mapping (SLAM) is one of the most important problems in robotics research, especially in the mobile robotics field. SLAM is concerned about accomplishing two tasks simultaneously: mapping an unknown environment with one or multiple mobile robots and localizing the mobile robot/robots. One common model of the environment consists of multiple landmarks such as objects, corners, visual features, salient points, etc. represented by points. And a coordinate vector is used to describe the location of each landmark in 2D or 3D space.

There are three main categories of methods for SLAM: the EKF SLAM, the particle filters related SLAM and the graph-based SLAM. The EKF SLAM [3–6] uses the extended Kalman filter [7, 8], which linearizes and approximates the originally nonlinear problem using the Jacobian of the model to get the system state vector and covariance matrix to be estimated and updated based on the environment measurements.

The graph-based SLAM [9–15] uses graph relationships to model the constraints on states of the landmarks and then uses nonlinear sparse optimization to solve the problem. The SLAM problem is modeled by a sparse graph, where the nodes represent the landmarks and each instant state, and edge or soft constraint between the nodes corresponds to either a motion or a measurement event. Based on high efficiency optimization methods that are mainly offline and the sparsity of the graph, graphical SLAM methods have the ability to scale to deal with much larger-scale maps.

The particle method for SLAM relies on particle filters [16], which enables easy representation for multimodal distributions since it is a non-parametric representation. The method uses particles representing guesses of true values of the states to approximate the posterior distributions. The first application of such method is introduced in [17]. The FastSLAM introduced in [18, 19] may be the most important and famous particle filter SLAM method.

However, each of these three methods has weaknesses and limitations:

For the EKF SLAM, the size of the system covariance matrix grows quadratically with the number of features or landmarks, thus heavy computation needs to be carried out in dense landmark environment. Such issue makes it unsuitable for processing large maps. Also, the linearization can cause inconsistency and divergence of the algorithm [1, 2].

For the graph-based SLAM, because performing the advanced optimization methods can be expensive, they are mostly not online. Moreover, the initialization can have a strong impact on the result.

Lastly, for the particle method for SLAM, a rigorous evaluation in the number of particles required is lacking; the number is often set manually relying on experience or trial and error. Second, the number of particles required increases exponentially with the dimension of the state space. Third, nested loops and extensive re-visits can lead to particles depletion, and make the algorithm fail to achieve a consistent map.

Our method generally falls into the category of Kalman filtering SLAM. By exploiting contraction analysis tools and virtual measurements, our algorithm in effect builds a stable linear time varying Kalman filter. Therefore, compared to the EKF SLAM methods, we do not suffer from errors brought by linearization process, and long term consistency is guaranteed. Also, because the Kalman filter is conditioned on the local coordinate attached to the robot, covariances between different landmarks are fully decoupled, which enables applications of the algorithm on large scale problems.

The Azimuth Model of the SLAM Problem

The model we use in this paper measures the azimuth angle in an inertial reference coordinate fixed to the center of the robot (Fig. 1), as in [20]. The robot is a point of mass with attitude and orientation.

The actual location of a lighthouse is described as $\mathbf{x} = (x_1, x_2)^T$ for 2D and (x_1, x_2, x_3) for 3D. The measured azimuth angle is $\theta = \arctan(\frac{x_1}{x_2})$. In 3D there is also the pitch measurement to the landmark $\phi = \arctan(\frac{x_3}{\sqrt{x_1^2+x_2^2}})$. Robot's translational velocity is $\mathbf{u} = (u_1, u_2)^T$ for 2D and $(u_1, u_2, u_3)^T$ for 3D. Ω is the angular velocity matrix of the robot: in 2D case, $\Omega = \begin{bmatrix} 0 & -\omega_z \\ \omega_z & 0 \end{bmatrix}$ and in 3D case,

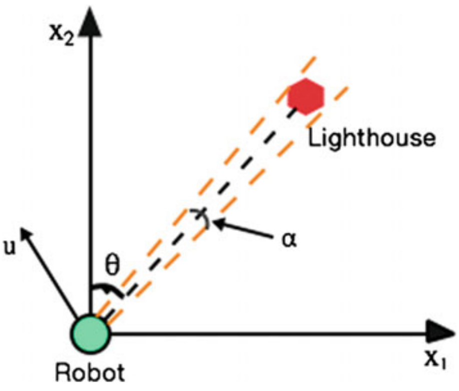


Fig. 1 Azimuth model

$\Omega = \begin{bmatrix} 0 & -\omega_z & \omega_y \\ \omega_z & 0 & -\omega_x \\ -\omega_y & \omega_x & 0 \end{bmatrix}$. Note here that in either case the matrix Ω is skew-symmetric. The range measurement from the robot to the landmark is $r = \sqrt{x_1^2 + x_2^2}$ for 2D and $r = \sqrt{x_1^2 + x_2^2 + x_3^2}$ for 3D. In our model, both $\mathbf{u}$ and Ω are assumed to be measured accurately, which is true in a lot of applications. So for any landmark in the inertial coordinate fixed to the robot, the relative motion is:

$$\dot{\mathbf{x}} = -\mathbf{u} - \Omega\mathbf{x}.$$

3 Basic Tools in Contraction Theory

Contraction theory [21] is a relatively recent dynamic analysis and design tool, which is an exact differential analysis of convergence of complex systems based on the knowledge of the system's linearization (Jacobian) at all points. Contraction theory converts a nonlinear stability problem into an LTV (linear time-varying) first-order stability problem by considering the convergence behavior of neighboring trajectories. While Lyapunov theory may be viewed as a "virtual mechanics" approach to stability analysis, contraction is motivated by a "virtual fluids" point of view. Historically, basic convergence results on contracting systems can be traced back to the numerical analysis literature [22–24].

Theorem [21] *Given the system equations* $\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, t)$, *where* $\mathbf{f}$ *is a differentiable nonlinear complex function of* $\mathbf{x}$ *within* C^n. *If there exists a uniformly positive definite metric* $\mathbf{M}$ *such that* Λ *is negative definite or*

$$\dot{\mathbf{M}} + \mathbf{M}\frac{\partial \mathbf{f}}{\partial \mathbf{x}} + \frac{\partial \mathbf{f}}{\partial \mathbf{x}}^{\mathbf{T}}\mathbf{M} \leq -\beta_M \mathbf{M}$$

with constant $\beta_M > 0$, *then all system trajectories converge exponentially to a single trajectory, which means contracting, with convergence rate* β_M.

Depending on the application, the metric can be found trivially (identity or rescaling of states), or obtained from physics (say, based on the inertia tensor in a mechanical system as e.g. in [25, 26]). For a summary of all features of contraction theory the reader can refer to [21].

4 Landmark Navigation and LTV Kalman Filter SLAM

In this section we illustrate the use of both LTV Kalman filter and contraction tools on the problem of navigation with visual measurements, an application often referred to as the landmark (or lighthouse) problem, and a key component of simultaneous localization and mapping (SLAM).

The main issues for EKF SLAM lies in the linearization and the inscalability caused by quadratic nature of the covariance matrix. If we can avoid these two issues, we could greatly improve EKF SLAM and provide exact optimal solutions to the SLAM problem. Moreover, this solution is suitable for large database applications as well.

Our approach to solve the SLAM problem in general follows the paradigms of LTV Kalman filter. And contraction analysis adds to the solution stability assurance because of the exponential convergence rate. In summary, our algorithm is a combination of both LTV Kalman filter and contraction analysis.

We present the results of an exact LTV Kalman observer based on the Riccati dynamics, which describes the Hessian of a Hamiltonian p.d.e. [20]. A rotation term similar to that of [27] in the context of perspective vision systems is also included.

4.1 *LTV Kalman Filter SLAM Using Virtual Measurements*

A standard extended Kalman Filter design [28] would start with the available nonlinear measurements, for example in 2D (Fig. 1),

$$\theta = \arctan(\frac{x_1}{x_2}) \quad \text{and/or} \quad r = \sqrt{x_1^2 + x_2^2}$$

and then linearize these measurements using the estimated Jacobian, leading to a locally stable observer. Intuitively, the starting point of our algorithm is the simple remark that the above relations can be equivalently written as

$$\mathbf{h}\mathbf{x} = \mathbf{0} \qquad \mathbf{h}^*\mathbf{x} = r$$

where

$$\mathbf{h} = (cos\theta, -sin\theta) \qquad \mathbf{h}^* = (sin\theta, cos\theta)$$

We exploit these exact linear time-varying expressions to achieve a globally stable observer design, and extend this idea in a variety of SLAM contexts.

Specifically, let us introduce the virtual, implicit measurement

$$\mathbf{y} = \mathbf{Hx} + \mathbf{v}(t)$$

where observation matrix $\mathbf{H}$ is a combination of measurement vectors $\mathbf{h}$ and $\mathbf{h}^*$. Since in the azimuth model, one has
In 2D

$$\mathbf{x} = (x_1, x_2)^T = (r\sin\theta, r\cos\theta)^T$$

In 3D

$$\mathbf{x} = (x_1, x_2)^T = (r\cos\phi\sin\theta, r\cos\phi\cos\theta, r\sin\phi)^T$$

this yields, for 2D scenarios, $\mathbf{h} = (cos\theta, -sin\theta)$, $\mathbf{h}^* = (sin\theta, cos\theta)$ as above, and for 3D scenarios:

$$\mathbf{h} = \begin{pmatrix} \cos\theta & -sin\theta & 0 \\ -\sin\phi\sin\theta & -\sin\phi\cos\theta & \cos\phi \end{pmatrix}$$

$$\mathbf{h}^* = (cos\phi sin\theta, cos\phi cos\theta, sin\phi)$$

All our propositions in the following cases have the same continuous LTV Kalman filter structure. The filter consists of two differential equations, one for the state estimate and one for the covariance:

$$\dot{\hat{\mathbf{x}}} = -\mathbf{u} - \Omega\hat{\mathbf{x}} + \mathbf{K}(\mathbf{y} - \mathbf{H}\hat{\mathbf{x}})$$

$$\dot{\mathbf{P}} = \mathbf{Q} - \mathbf{PH^TR^{-1}HP} - \Omega\mathbf{P} - \mathbf{P}\Omega^T$$

where the Kalman gain is given by

$$\mathbf{K} = \mathbf{PH^TR^{-1}}$$

and the other terms are defined as:

$$\mathbf{Q} = cov(\mathbf{u} + \Omega\hat{\mathbf{x}}) \qquad \mathbf{y} = \mathbf{Hx} + \mathbf{v(t)} \qquad \mathbf{v(t)} \sim \mathbf{N(0, R)}$$

where $\mathbf{y}$ is the measurement or observation vector, which includes both actual and virtual measurements, and $\mathbf{H}$ is the observation matrix. $\mathbf{v}(t)$ is a zero-mean white noise vector with the covariance $\mathbf{R}$. In each of our case presented in the later section, the LTV Kalman filter structure won't be repetitively introduced, while we will explain the specific virtual measurement $\mathbf{y}$ and corresponding observation model $\mathbf{H}$ in each case.

Case I: Bearing Information Only

This original version of bearing-only SLAM was originally presented in [20]. Physically, the virtual measurement error term $\mathbf{h}\hat{\mathbf{x}}$ corresponds to rewriting an angular error as a tangential position error between estimated and true landmark. The system contracts in the tangential direction if $\mathbf{R}^{-1} > 0$, and it is indifferent along the unmeasured radial direction.

Case II: Radial Contraction with Independent $\dot{\theta}$ Information

In this case we utilize $\dot{\theta}$ as additional information. $\dot{\theta}$ is the relative angular velocity from the robot to the landmark and we also have $\dot{\phi}$ in the 3D case. Independent $\dot{\theta}$ measurement could be achieved either computationally based on θ or through optical flow algorithms on visual sensors. We propose here that $\dot{\theta}$ gives us an additional dimension of information that help the LTV Kalman filter with radial contraction. The additional constraint or observation we get is based on the relationship that

$$\textit{radial distance} \times \textit{angular velocity} = \textit{tangential velocity}$$

where in our case $\mathbf{h}^*\hat{\mathbf{x}}$ is the length of vector $\hat{\mathbf{x}}$ projected on the direction along azimuth direction to represent the estimated range. So if the estimation is precise, $\dot{\theta}\mathbf{h}^*\hat{\mathbf{x}} + \mathbf{h}\Omega\hat{\mathbf{x}}$ should equal to $-\mathbf{hu}$, which is the relative velocity projected along the tangential direction.

In this case, $\mathbf{y}_1 = \mathbf{hx} = \mathbf{0}$ is the constraint on bearing measurement and $\mathbf{y_2} = (\dot{\theta}\mathbf{h}^* + \mathbf{h}\Omega)\mathbf{x} = -\mathbf{hu}$ in 2D or $\mathbf{y_2} = (\begin{bmatrix} \dot{\theta}\mathbf{h}^* \\ \dot{\phi}\mathbf{h}^* \end{bmatrix} + \mathbf{h}\Omega)\mathbf{x} = -\mathbf{hu}$ in 3D is the constraint about bearing velocity, radial distance and the tangential velocity.

So the virtual measurement is consisted of two parts:

virtual measurement:
$$\mathbf{y} = \begin{bmatrix} \mathbf{y_1} \\ \mathbf{y_2} \end{bmatrix} = \begin{bmatrix} \mathbf{0} \\ -\mathbf{hu} \end{bmatrix}$$

observation model: (2D) $H = \begin{bmatrix} \mathbf{h} \\ \dot{\theta}\mathbf{h}^* + \mathbf{h}\Omega \end{bmatrix}$ (3D) $H = \begin{bmatrix} \mathbf{h} \\ \begin{bmatrix} \dot{\theta}\mathbf{h}^* \\ \dot{\phi}\mathbf{h}^* \end{bmatrix} + \mathbf{h}\Omega \end{bmatrix}$.

Case III: With Time to Contact Measurement τ

In this case we utilize the "time to contact" measurement as additional information. Time-to-contact [29] measurement provides an estimation of time to reach the lighthouse, which could suggest the radial distance to the lighthouse based on local velocity information. This is one popular measurement for sailing and also utilized by animals and insects. For a robot, the "time to contact" measurement could be potentially achieved by optical flows algorithms or some novel sensors specifically developed for that.

$$\tau = |\frac{\alpha}{\dot{\alpha}}| \approx |\frac{r}{\dot{r}}|$$

As shown in Fig. 1, we can get the measurement $\tau = |\frac{\alpha}{\dot{\alpha}}|$, where α is an small angle measured between two feature points, edges on a single distant landmark for example. In our case, we use the angle between two edges of the cylinder landmark so that $\alpha \approx arctan(\frac{d}{r})$, where d is the diameter of the cylinder landmark and r is the distance from the robot to the landmark. One thing to notice is that the time-to-contact measurement is an approximation. Also when $\mathbf{uh} \approx 0$, τ would be reaching infinity, which reduces the reliability of the algorithm near that region. Thus in this case besides the bearing constraint $\mathbf{y}_1 = \mathbf{hx} = \mathbf{0}$, we propose a novel constraint y_3 utilizing the "time to contact" τ.

As we know $r = \mathbf{h}^*\mathbf{x}$ so that

$$\dot{r} = -\mathbf{h}^*\mathbf{u} - \mathbf{h}^*\Omega\mathbf{x} + \dot{\mathbf{h}}^*\mathbf{x}$$

Since $\mathbf{h}^*$ is the unit vector with the same direction of $\mathbf{x}$, both $\mathbf{h}^*\Omega\mathbf{x}$ and $\dot{\mathbf{h}}^*x$ equal to 0, so simply $\dot{r} = -\mathbf{h}^*\mathbf{u}$, and

$$\tau = |\frac{r}{\dot{r}}| = \frac{\mathbf{h}^*\mathbf{x}}{|-\mathbf{h}^*\mathbf{u}|}$$

which means: $|\tau\mathbf{h}^*\mathbf{u}| \approx \mathbf{h}^*\mathbf{x}$ so we can have $\mathbf{y_3} = |\tau\mathbf{h}^*\mathbf{u}|$

$$\mathbf{y} = \begin{bmatrix} \mathbf{y_1} \\ \mathbf{y_3} \end{bmatrix} = \begin{bmatrix} \mathbf{0} \\ |\tau\mathbf{h}^*\mathbf{u}| \end{bmatrix} \qquad \mathbf{H} = \begin{bmatrix} \mathbf{h} \\ \mathbf{h}^* \end{bmatrix}$$

So that $\mathbf{y} = \mathbf{Hx} + \mathbf{v}$, and it is applicable to both 2D and 3D cases.

Case IV: With Radial Measurement

If we have both bearing measurement θ and ϕ and radial measurement r, the new constraint would be $y_4 = r = \mathbf{h}^*\mathbf{x}$.

So that for both 2D and 3D the virtual measurement is:

$$\mathbf{y} = \begin{bmatrix} \mathbf{y_1} \\ \mathbf{y_4} \end{bmatrix} = \begin{bmatrix} \mathbf{0} \\ r \end{bmatrix} \qquad \mathbf{H} = \begin{bmatrix} \mathbf{h} \\ \mathbf{h}* \end{bmatrix}$$

The landmark positions that we estimate here are based on the azimuth model in the inertial coordinate system fixed to the robot. So the positions of the landmarks are actually relative positions to the robot rather than global locations. And what we are doing here is mainly mapping the local surrounding landmarks, so instead of generally naming the states $\mathbf{x}$, recognizing them by $\mathbf{x}_{il}$ is more appropriate, with corresponding measurements θ_i and r_i, each with independent covariance matrix P_i.

$$\dot{\hat{\mathbf{x}}}_{il} = -\mathbf{u} - \Omega\hat{\mathbf{x}}_{\mathbf{il}} + \mathbf{K}(\mathbf{y} - \mathbf{H}_{\mathbf{il}}\hat{\mathbf{x}}_{\mathbf{il}})$$

$$\dot{\mathbf{P}}_{\mathbf{i}} = \mathbf{Q} - \mathbf{P_i}\mathbf{H}_{\mathbf{il}}^{\mathbf{T}}\mathbf{R}^{-1}\mathbf{H}_{\mathbf{il}}\mathbf{P_i} - \Omega\mathbf{P_i} - \mathbf{P_i}\Omega^T.$$

Transform to Global Coordinates

When we try to transform the local observations to global coordinates recovering both the map and the location of the vehicle, we need to consider the robot heading β. We use the local estimations $\mathbf{x}_{il}$'s in the first stage as inputs to the second stage. Remember that we have the coordinates transformation for each landmark $\mathbf{x_i}$ (global) and the vehicle position $\mathbf{x_v}$ as:

$$\mathbf{x_i} - \mathbf{x_v} = \begin{bmatrix} cos\beta & sin\beta \\ -sin\beta & cos\beta \end{bmatrix} \begin{bmatrix} x_{il1} \\ x_{il2} \end{bmatrix}$$

which we can transform and hence use $\mathbf{x}_\beta = \begin{bmatrix} cos\beta \\ sin\beta \end{bmatrix}$ as new states related to the heading of the vehicle, where $\mathbf{x}_\beta^{\mathbf{T}}\mathbf{x}_\beta = 1$. So that once again we have linear constraint as

$$\mathbf{x_i} - \mathbf{x_v} = \begin{bmatrix} x_{il1} & x_{il2} \\ x_{il2} & -x_{il1} \end{bmatrix} \begin{bmatrix} cos\beta \\ sin\beta \end{bmatrix} = H_i \mathbf{x}_\beta$$

So we can use an LTV Kalman-like system updated as:

$$\frac{d}{dt}\begin{bmatrix} \mathbf{x_1} \\ \mathbf{x_2} \\ \vdots \\ \mathbf{x_n} \\ \mathbf{x_v} \\ \mathbf{x}_\beta \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ \mathbf{u} \\ \begin{bmatrix} 0 & -\omega \\ \omega & 0 \end{bmatrix} \mathbf{x}_\beta \end{bmatrix} + PH^T R^{-1} \left(\begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ 1 \end{bmatrix} - \begin{bmatrix} I & 0 & \cdots & 0 & -I & -H_1 \\ 0 & I & \cdots & 0 & -I & -H_2 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \cdots & I & -I & -H_n \\ 0 & 0 & \cdots & 0 & 0 & \mathbf{x}_\beta^{\mathbf{T}} \end{bmatrix} \begin{bmatrix} \mathbf{x_1} \\ \mathbf{x_2} \\ \vdots \\ \mathbf{x_n} \\ \mathbf{x_v} \\ \mathbf{x}_\beta \end{bmatrix} \right)$$

Covariance updates

$$\dot{\mathbf{P}} = \mathbf{Q} - \mathbf{PH^T R^{-1} HP} - \Omega\mathbf{P} - \mathbf{P}\Omega^T$$

where the skew-symmetric matrix

$$\Omega = \begin{bmatrix} \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \begin{bmatrix} 0 & -\omega \\ \omega & 0 \end{bmatrix} \end{bmatrix}$$

Here only the $\mathbf{x}_\beta^{\mathbf{T}}\mathbf{x}_\beta = 1$ is nonlinear. All the remaining constraints of the system are all time varying linear constraints. Note that in this stage of transforming local estimations to global coordinates, we are actually utilizing a full state Kalman filter with results from the first stage as virtual inputs. Computationally, the LTV Kalman filter at this stage takes as much computation as the traditional EKF methods. The differences are: first, our LTV Kalman filter is mostly linear except for the part

$\mathbf{x}_\beta^\mathbf{T}\mathbf{x}_\beta = 1$, and it is exact; second, our LTV Kalman filter can solve problems where radial measurements are not available.

Remark I: Nonlinearity in Vehicle Kinematics

When traditional EKF SLAM methods are applied to ground vehicles, another nonlinearity arises from the vehicle kinematics. This is easily incorporated in our model. For the most general case, the vehicle motion can be modeled as

$$\dot{x}_{v1} = ucos\beta \qquad \dot{x}_{v2} = usin\beta \qquad \dot{\beta} = \frac{u}{L}tan\theta_s = \omega$$

where u is the linear velocity, L is the distance between the front and rear axles and θ_s is the steering angle. Since in our case, we use $cos\beta$ and $sin\beta$ as states to estimate instead of β, our LTV Kalman filter can be based on the linear form:

$$\frac{d}{dt}\begin{bmatrix} x_{v1} \\ x_{v2} \\ cos\beta \\ sin\beta \end{bmatrix} = \begin{bmatrix} 0 & 0 & u & 0 \\ 0 & 0 & 0 & u \\ 0 & 0 & 0 & -\omega \\ 0 & 0 & \omega & 0 \end{bmatrix}\begin{bmatrix} x_{v1} \\ x_{v2} \\ cos\beta \\ sin\beta \end{bmatrix}$$

Remark II: Reduced Order Observer

When transforming to global coordinates, we used a full order Kalman filter in the previous section. However, since Sect. 4.1 already provides the positions of the landmarks relative to the vehicle, one could potentially reduce computational cost by developing reduced observers to estimate only the vehicle's position and pose, and estimate global location of the landmarks based on the vehicle's trajectory.

4.2 Contraction Analysis for the LTV Kalman Filter

Since all our four cases follow the same LTV Kalman filter structure, we can analyze the contraction property in general for all four cases at the same time. The LTV Kalman filter system we proposed previously contracts according to Sect. 3, with metric $\mathbf{M_i} = \mathbf{P_i}^{-1}$, as analyzed in [20]:

$$\frac{\partial \mathbf{f_i}}{\partial \mathbf{x_i}}^T \mathbf{M_i} + \mathbf{M_i}\frac{\partial \mathbf{f_i}}{\partial \mathbf{x_i}} + \dot{\mathbf{M}}_\mathbf{i} = -\mathbf{M_i}\mathbf{Q_i}\mathbf{M_i} - \mathbf{H_{il}^T}\mathbf{R}^{-1}\mathbf{H_{il}}$$

The above leads to the global exponential Kalman observer of landmarks(lighthouses) around a vehicle. Hence for any initial value, our estimation will converge to the trajectory of the true landmarks positions exponentially. It gives stability proof to the proposed LTV Kalman filter and boundedness of M is given with the observability gramian. However, LTV Kalman cannot compute the convergence rates explicitly, because the convergence rate is given by the eigenvalues of $-\mathbf{M_i Q_i M_i} - \mathbf{H_{il}^T R}^{-1}\mathbf{H_{il}}$ which is related to M.

This system is contracting with metric $M = P^{-1}$ at the second stage also. Since the second stage only use the results of the first stage as pure inputs, and both stages are contracting, according to the hierarchical combination of the contraction analysis [21], the whole system consisted of two stages is contracting. Since the true locations of landmarks and path of the vehicle are particular solutions to the system, all trajectories of the state vectors would converge exponentially to the truth.

4.3 Noise Analysis

The basic assumption for the Kalman filter is that the noise signal $\mathbf{v}(t) = \mathbf{y} - \mathbf{Hx}$ is a zero-mean Gaussian noise. Since the actual measurements that we obtain from a robot are θ, ϕ,$\dot{\theta}$,$\dot{\phi}$,r, and τ, we need to check that the mean error remains zero after incorporating them into the virtual measurements.

Consider the bearing angle θ we measure comes with a zero-mean white Gaussian noise $w \sim N(0, \sigma_\theta^2)$, and ϕ with zero-mean Gaussian noise $N(0, \sigma_\phi^2)$ then

$$E[cos(z)] = E[cos(\theta + w)] = e^{-\frac{\sigma_\theta^2}{2}} cos(\theta)$$

$$E[sin(z)] = E[sin(\theta + w)] = e^{-\frac{\sigma_\theta^2}{2}} sin(\theta)$$

For our virtual measurement $\mathbf{y} = \mathbf{hx}$ where $\mathbf{h} = [cos(\theta), -sin(\theta]$, the error $v = 0 - \mathbf{hx} = -(x_1 cos(\theta) - x_2 sin(\theta))$, so that the mean of the error

$$E[v] = -e^{-\frac{\sigma_\theta^2}{2}}(x_1 cos(\theta) - x_2 sin(\theta)) = 0$$

which means there is no bias in this case.

In the 3D case $\mathbf{h} = \begin{pmatrix} \cos\theta & -sin\theta & 0 \\ -\sin\phi\sin\theta & -\sin\phi\cos\theta & \cos\phi \end{pmatrix}$

$$E[\mathbf{v}] = -\begin{pmatrix} e^{-\frac{\sigma_\theta^2}{2}}(x_1 cos(\theta) - x_2 sin(\theta)) \\ e^{-\frac{\sigma_\theta^2}{2}-\frac{\sigma_\phi^2}{2}}(-x_1 sin(\phi)sin(\theta) - x_2 sin(\phi)cos(\theta)) + e^{-\frac{\sigma_\phi^2}{2}} cos(\phi)x_3 \end{pmatrix}$$

$$E[\mathbf{v}] = \begin{pmatrix} 0 \\ -\left(e^{-\frac{\sigma_\phi^2}{2}} - e^{-\frac{\sigma_\theta^2+\sigma_\phi^2}{2}}\right) cos(\phi)x_3 \end{pmatrix}$$

So there would be a bias in the second term.

A bias in the mean error is generally caused by the product of to trigonometric functions about ϕ or θ, which only happens in the 3D cases. It would bring in an extra scale factor of $e^{-\frac{\sigma_\theta^2}{2}}$, which unbalances the original equation.

Following the same logic and process, we get that in Case II(3D)

$$E[\mathbf{v}] = \begin{pmatrix} 0 \\ (e^{-\frac{\sigma_\phi^2}{2}} - e^{-\frac{\sigma_\theta^2+\sigma_\phi^2}{2}})(-x_1 sin(\phi)sin(\theta) - x_2 sin(\phi)cos(\theta)) \\ \dot{\theta}(e^{-\frac{\sigma_\phi^2}{2}} - e^{-\frac{\sigma_\theta^2+\sigma_\phi^2}{2}})(x_1 cos(\phi)sin(\theta) + x_2 cos(\phi)cos(\theta)) \\ (e^{-\frac{\sigma_\phi^2}{2}} - e^{-\frac{\sigma_\theta^2+\sigma_\phi^2}{2}})cos(\phi)(\dot{\phi}(x_1 sin(\theta) + x_2 cos(\theta)) - u_1 sin(\theta) - u_2 cos(\theta)) \end{pmatrix}$$

In Case III(3D),

$$E[\mathbf{v}] = \begin{pmatrix} 0 \\ (e^{-\frac{\sigma_\phi^2}{2}} - e^{-\frac{\sigma_\theta^2+\sigma_\phi^2}{2}})(-x_1 sin(\phi)sin(\theta) - x_2 sin(\phi)cos(\theta)) \\ (e^{-\frac{\sigma_\phi^2}{2}} - e^{-\frac{\sigma_\theta^2+\sigma_\phi^2}{2}})cos(\phi)(x_1 sin(\theta) + x_2 cos(\theta) + |\tau(u_1 sin(\theta) + u_2 cos(\theta))| \end{pmatrix}$$

In Case IV(3D),

$$E[\mathbf{v}] = \begin{pmatrix} 0 \\ (e^{-\frac{\sigma_\phi^2}{2}} - e^{-\frac{\sigma_\theta^2+\sigma_\phi^2}{2}})(-x_1 sin(\phi)sin(\theta) - x_2 sin(\phi)cos(\theta)) \\ (e^{-\frac{\sigma_\phi^2}{2}} - e^{-\frac{\sigma_\theta^2+\sigma_\phi^2}{2}})(x_1 cos(\phi)sin(\theta) + x_2 cos(\phi)cos(\theta)) \end{pmatrix}$$

We can see that in each case, the mean errors only shift in the 3D cases, and with a scale coefficient of $e^{-\frac{\sigma_\phi^2}{2}} - e^{-\frac{\sigma_\theta^2+\sigma_\phi^2}{2}}$. When the variances σ_θ and σ_ϕ are small, that coefficient is almost zero. Even when we increase in simulations the actual variances of the bearing measurements to 10° (which is unrealistic based on the performances of current instruments), the mean shift is still in on the scale of 10^{-2}m, and thus remains negligible. So we would suggest numerically that the noise distribution of the actual measurements does not matter to compute (and subtract) the mean.

5 Experiments

Experiments for 2D Landmarks Estimation

We experiment the 2D version of our cases with simulations in Matlab. As shown in [30], in the simulations, we have three lighthouses with different locations. The diameter of each landmark is $d = 2$ m. We have run simulations on all four cases. The noise signals that we use in the simulations are: standard variance for zero-mean Gaussian noise of θ is 2°; standard variance for noise of $\dot{\theta}$ is $5\,°/s$; standard variance

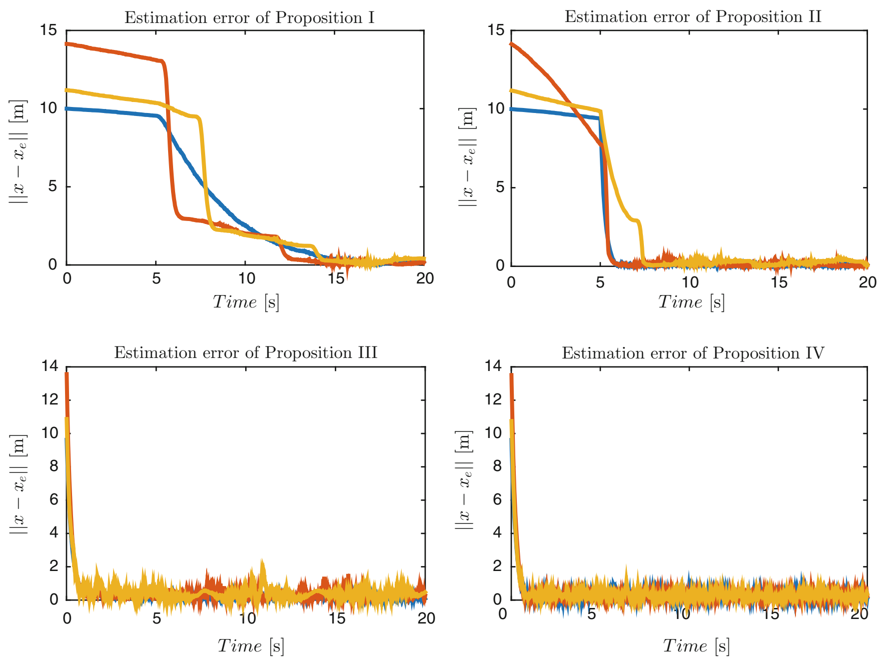


Fig. 2 Case II, III, IV and the original Case I for 2D estimation

for measurement noise of r is $2m$; and standard variance for noise of α is 0.5°. In [30], the red lines indicate the trajectories of estimations of the Case I(original), which are used as references for all other three. The green lines are the trajectory of Case II, III, and IV. The blue lines are the movement trajectory of the vehicle. As shown in the diagram, trajectories of estimations from Case II, III, and IV are smoother and more directed than the original Case I. This is because the trajectory exploits additional information. In particular, for Case III and IV, since the "time-to-contact" measurement and radial distance measurement both contains information on the radial direction, they converge to the true position directly, without waiting for the vehicle movement to bring in extra information.

Next, we analyze the estimation errors $||\mathbf{x} - \hat{\mathbf{x}}||$ for all three lighthouses in Fig. 2. The figure shows that the errors decay faster in Case II, III, IV. The difference is that Case II needs to wait for the movement of the vehicle to provide more information about $\dot{\theta}$, yet Case III and IV contract much faster with exponential rates because of radial related measurements. Compared to Case IV, Case III is less smooth, as expected, because the "time-to-contact" measurement itself is an approximation and may be disturbed when $\dot{r}$ is close to zero.

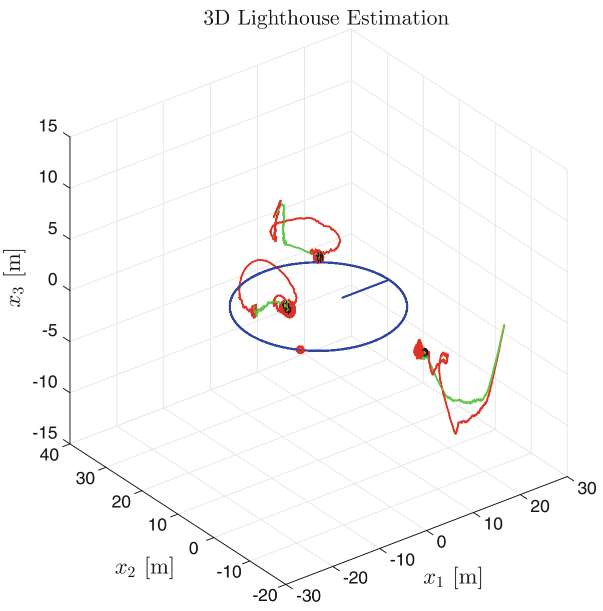


Fig. 3 3D landmarks estimation in Case I and II

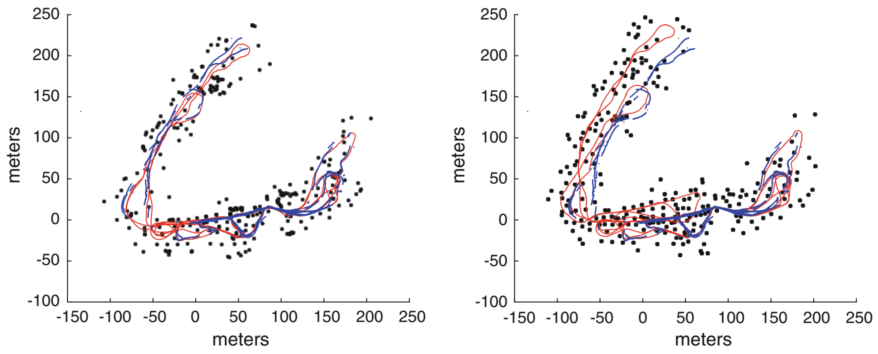


Fig. 4 On the *left* is the path and landmarks estimation of our algorithm and on the *right* is the result from Unscented Fast SLAM. The thick *blue* path is the GPS data and the solid *red* path is the estimated path; the *black* asterisks are the estimated positions of the landmark

Experiments for 3D Landmarks Estimation

We also have simulation results for Case I and Case II in 3D settings. Here we also have three lighthouses with different locations. The results shown in Fig. 3 suggest that our algorithm is capable of estimating landmarks positions accurately in 3D space with bearing angle for both yaw and pitch. Animations of all simulation results are provided at [30].

Experiment for Victoria Park Landmarks Estimation

We applied our algorithm to Sydney Victoria Park dataset, a popular dataset in the SLAM community. The vehicle path around the park is about 30 min, covering over 3.5 km. Landmarks in the park are mostly trees. Estimation results are compared with intermittent GPS information as ground truth to validate the states of the filters as shown in Fig. 4. Our estimated track compares favorably to benchmark result of [31], which highlights the consistency of our algorithm in large scale applications. Simulation result of the Victoria Park dataset is provided at [32].

6 Concluding Remarks

In this paper, we propose using the combination of LTV Kalman filter and contraction tools to solve the problem of simultaneous mapping and localization (SLAM). By exploiting the virtual measurements, the LTV Kalman observer does not suffer from errors brought by the linearization process in the EKF SLAM. And conditioned on the robot position, the covariances between landmarks are fully decoupled, which makes the algorithm possible to be scaled to solve large datasets. Contraction analysis provides proof on stability of the algorithm and the contracting rates. The series of application cases using proposed algorithms utilize different kinds of sensor information that range from traditional bearing measurements and radial measurements to novel ones like optical flows and time-to-contact measurements. They can solve SLAM problems in both 2D and 3D scenarios. Note that

- the multi-dimensional landmark navigation task corresponds to feature-based SLAM, where features are extracted and followed in a stream. These features as e.g. tree or a stone can be regarded as a landmark.
- bounding of **P** is an analytic computation of the observability gramian in [28].
- the novelty of the propositions is that by exploiting the virtual measurements, the algorithm corresponds to an exact and global linear observer design.
- all landmark observers are fully decoupled for the local estimation. In SLAM the state-covariance or the information matrix may be sparse, but are not fully decoupled. However if we want to estimate a smooth surface (e.g. a wall) then we can add in smoothing terms to couple them as suggested in [20].
- our algorithm is particularly effective when distance measurements are not available, with good quality vision sensors becoming available at very low cost compared to range sensors like lidars.
- it may also be interesting to consider whether similar representations may also be used in biological navigation, e.g. in the context of place cells or grid cells [33].

References

1. Huang, S., Dissanayake, G.: Convergence and consistency analysis for extended kalman filter based SLAM. IEEE Trans. Robot. **23**(5), 1036–1049 (2007)
2. Bailey, T., Nieto, J., Guivant, J., Stevens, M., Nebot, E.: Consistency of the ekf-slam algorithm. In: 2006 IEEE/RSJ International Conference on Intelligent Robots and Systems, pp. 3562–3568. IEEE (2006)
3. Moutarlier, P., Chatila, R.: An experimental system for incremental environment modelling by an autonomous mobile robot. In: Experimental Robotics I, pp. 327–346. Springer (1990)
4. Cheeseman, P., Smith, R., Self, M.: A stochastic map for uncertain spatial relationships. In: 4th International Symposium on Robotic Research, pp. 467–474 (1987)
5. Smith, R., Self, M., Cheeseman, P.: Estimating uncertain spatial relationships in robotics. In: Autonomous Robot Vehicles, pp. 167–193. Springer (1990)
6. Moutarlier, P., Chatila, R.: Stochastic multisensory data fusion for mobile robot location and environment modeling. In: International Symposium of Robotics Research (1989)
7. Jazwinski, A.H.: Stochastic Processes and Filtering Theory. Courier Corporation, New York (2007)
8. Rudolph Emil Kalman: A new approach to linear filtering and prediction problems. J. Fluids Eng. **82**(1), 35–45 (1960)
9. Konolige, K.: Large-scale map-making. In: Proceedings of the National Conference on Artificial Intelligence, pp. 457–463. AAAI Press, Menlo Park (1999), MIT Press, Cambridge (2004)
10. Montemerlo, M., Thrun, S.: Large-scale robotic 3-d mapping of urban structures. In: Experimental Robotics IX, pp. 141–150. Springer (2006)
11. Grisetti, G., Kummerle, R., Stachniss, C., Burgard, W.: A tutorial on graph-based SLAM. IEEE Intell. Trans. Syst. Mag. **2**(4), 31–43 (2010)
12. Folkesson, J., Christensen, H.: Graphical slam-a self-correcting map. In: Proceedings of the 2004 IEEE International Conference on Robotics and Automation, ICRA 2004, vol. 1, pp. 383–390. IEEE (2004)
13. Duckett, T., Marsland, S., Shapiro, J.: Fast, on-line learning of globally consistent maps. Auton. Robots **12**(3), 287–300 (2002)
14. Dellaert, F., Kaess, M.: Square root sam: simultaneous localization and mapping via square root information smoothing. Int. J. Robot. Res. **25**(12), 1181–1203 (2006)
15. Thrun, S., Montemerlo, M.: The graph SLAM algorithm with applications to large-scale mapping of urban structures. Int. J. Robot. Res. **25**(5–6), 403–429 (2006)
16. Matthies, L., Shafer, S.A.: Error modeling in stereo navigation. IEEE J. Robot. Autom. **3**(3), 239–248 (1987)
17. Doucet, A., De Freitas, N., Murphy, K., Russell, S.: Rao-blackwellised particle filtering for dynamic bayesian networks. In: Proceedings of the Sixteenth conference on Uncertainty in Artificial Intelligence, pp. 176–183. Morgan Kaufmann Publishers Inc., San Francisco (2000)
18. Montemerlo, M., Thrun, S., Koller, D., Wegbreit, B. et al.: Fastslam: a factored solution to the simultaneous localization and mapping problem. In: AAAI/IAAI, pp. 593–598 (2002)
19. Montemerlo, M., Thrun, S.: Fastslam 2.0. FastSLAM: A Scalable Method for the Simultaneous Localization and Mapping Problem in Robotics, pp. 63–90 (2007)
20. Lohmiller, W., Slotine, J.-J.: Contraction analysis of nonlinear hamiltonian systems. In: 2013 IEEE 52nd Annual Conference on Decision and Control (CDC), pp. 6586–6592. IEEE (2013)
21. Lohmiller, W., Slotine, J.-J.E.: On contraction analysis for non-linear systems. Automatica **34**(6), 683–696 (1998)
22. Lewis, D.C.: Metric properties of differential equations. Am. J. Math. **71**(12), 294–312 (1949)
23. Hartman, P.: Ordinary Differential Equations. Wiley, New York (1964)
24. Demidovich, B.P.: Dissipativty of a system of nonlinear differential equations. Ser. Mat. Mekh. **6**, 19–27 (1961)
25. Lohmiller, W., Slotine, J.J.E.: Shaping state-dependent convergence rates in nonlinear control system design. In: AIAA Guidance, Navigation, and Control Conference (2008)

26. Lohmiller, W., Slotine, J.J.E.: Exact decomposition and contraction analysis of nonlinear hamiltonian systems. In: AIAA Guidance, Navigation, and Control Conference (2013)
27. Grave, I., Tang, Y.: A new observer for perspective vision systems under noisy measurements. IEEE Trans. Autom. Control **60**(2), 503–508 (2015)
28. Bryson, A.E.: Applied optimal control: optimization, estimation and control. CRC Press, Boca Raton (1975)
29. Browning, A.N.: A neural circuit for robust time-to-contact estimation based on primate mst. Neural Comput. **24**(11), 2946–2963 (2012)
30. https://vimeo.com/channels/910603
31. Kim, C., Sakthivel, R., Chung, W.K.: Unscented fastslam: a robust and efficient solution to the slam problem. IEEE Trans. Robot. **24**(4), 808–820 (2008)
32. https://vimeo.com/136219156
33. Moser, E.I., Kropff, E., Moser, M.-B.: Place cells, grid cells, and the brain's spatial representation system. Annu. Rev. Neurosci. **31**, 69–89 (2008)

Exploiting Photometric Information for Planning Under Uncertainty

Gabriele Costante, Jeffrey Delmerico, Manuel Werlberger, Paolo Valigi and Davide Scaramuzza

1 Introduction

We consider the problem of planning an optimal trajectory between two spatial locations in an initially unknown environment with an autonomous, vision-controlled, micro aerial vehicle (MAV). In many previous works, optimal trajectories are those with the shortest or lowest effort path to the goal position. To improve the performance of vision-based control, and consequently all of the other perception functions that rely on the robot's pose estimate, we instead consider optimal trajectories to be those that minimize the uncertainty in this pose estimate. Because we compute the robot pose uncertainty as a function of the photometric information of the scene, we call this approach *Perception-aware Path Planning*.

Despite the impressive results achieved in visual SLAM applications [1, 2], most of vision-controlled MAVs navigate towards a goal location using a predefined set of viewpoints or by remote control, without responding to environmental conditions [3, 4]. Recently, several works have tackled the problem of autonomously planning an optimal trajectory towards a goal location [5, 6], and others have extended this to uncertainty-aware planning that tries to provide high localization accuracy [7, 8].

G. Costante (✉) · P. Valigi
University of Perugia, Perugia, Italy
e-mail: gabriele.costante@unipg.it

P. Valigi
e-mail: paolo.valigi@unipg.it

J. Delmerico · M. Werlberger · D. Scaramuzza
University of Zürich, Zürich, Switzerland
e-mail: jeffdelmerico@ifi.uzh.ch

M. Werlberger
e-mail: werlberger@ifi.uzh.ch

D. Scaramuzza
e-mail: sdavide@ifi.uzh.ch

A. Bicchi and W. Burgard (eds.), *Robotics Research*, Springer Proceedings in Advanced Robotics 2, DOI 10.1007/978-3-319-51532-8_7

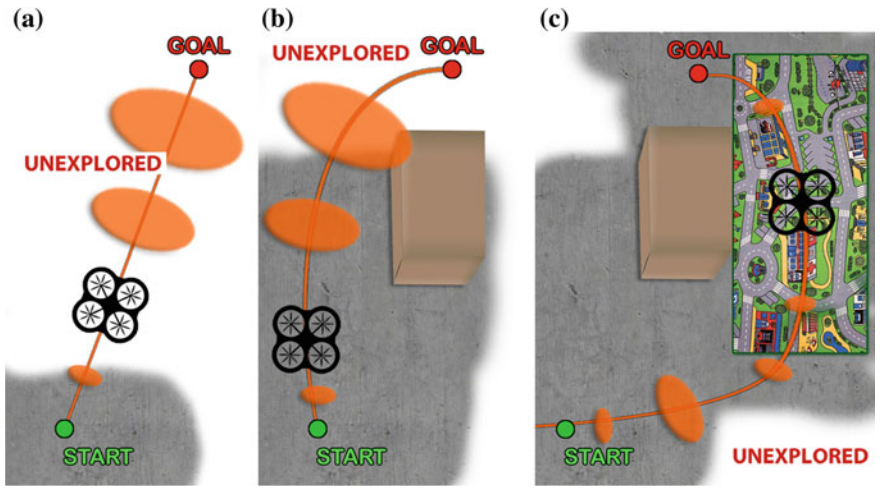


Fig. 1 Online perception-aware path planning: An initial plan is computed without prior knowledge about the environment (**a**). The plan is then updated as new obstacles (**b**) or new textured areas (**c**) are discovered. Although the new trajectory is longer than the one in **b**, it contains more photometric information and, thus, is optimal with respect to the visual localization uncertainty

However, these approaches discard the photometric information (i.e, texture) of the scene and plan the trajectory in advance, which requires prior knowledge of the full 3D map of the environment. We propose a system that instead selects *where to look*, in order to capture the maximum visual formation from the scene to ensure pose estimates with low uncertainty.

Additionally, we consider the scenario where the robot has no prior knowledge of the environment, and it explores to generate a map and navigate to a goal. Without an a priori map, we update the planned path as new images are collected by the camera while the robot explores the surroundings (see Fig. 1). In particular, we utilize the photometric information in the newly observed regions of the environment to determine the optimal path with respect to pose uncertainty. To the best of our knowledge, this is among the first works that propose to plan a *perception-aware* trajectory *on-the-fly*, while perceiving the environment with only a camera sensor.

We evaluate the proposed methods with several different experiments designed to illustrate the feasibility of our approach for an autonomous MAV, and to demonstrate the improvement in pose uncertainty when planning with perception awareness.

1.1 Related Work

When the minimization of the localization uncertainty is considered in the planning process, the problem is often referred to as "Planning under Uncertainty" or "Planning in Information Space". Probabilistic planning with the inability to directly observe all the state information is often based on Partially Observable Markov Decision

Processes (POMDPs) or solved as a graph-search problem. The major drawback of these approaches is their exponential growth in computational complexity. Sim and Roy [9] selected trajectories that maximize the map reconstruction accuracy in SLAM applications. They proposed to use a breadth-first search over possible robot positions to predict a sequence of EKF estimates and select the one that lead to the maximum information gain. Recently, sampling-based methods have been introduced to plan trajectories in complex configuration spaces. Optimal Rapidly-exploring Random Trees (RRT*s) [10] have been widely used in path planning problems and their extension to Rapidly-exploring Random Belief Trees (RRBTs) [7] takes pose uncertainty into account and avoids collisions.

Selecting sequences of viewpoints that optimize for a certain task (e.g, pose estimation or map uncertainty minimization) is referred to as *active perception* [11, 12]. While previous papers on active perception relied on using range sensors (e.g, [8]), Davison and Murray [13] were among the first to use vision sensors (a stereo camera setup) to select where the camera *should look* to reduce the pose drift during visual SLAM. More recently, Sadat et al. [14] and Achtelik et al. [15] investigated optimal path planning by leveraging visual cues. The former ensures good localization accuracy by extending RRTs* to select feature-rich trajectories, while the latter uses RRBT to compute the propagation of the pose uncertainty by minimizing the reprojection error of 3D map points. Kim and Eustice [16] proposed a Perception Driven Navigation (PDN) framework: the robot follows a pre-planned path and computes information gain at the viewpoints along it, but revisits already-explored, highly-salient areas to regain localization accuracy if its pose uncertainty increases.

It should be noted that all approaches mentioned so far [13–16] rely on *sparse 2D features* to compute highly-informative trajectories. By contrast, in this paper we rely on *direct methods* [17]. Contrarily to feature-based approaches—which only use small patches around corners—direct methods use all information in the image, including edges. They have been shown to outperform feature-based methods in terms of robustness and accuracy in sparsely-texture scenes [1, 2, 18].

Several works have addressed the problem of online planning. Efficient replanning was addressed in Ferguson et al. [19] by updating the trajectory whenever a new obstacle is spotted. For RRT*, Boardman et al. [20] proposed to dynamically update an initial planned trajectory by computing a new RRT* tree rooted by the robot's current location and reusing branches from the initially-grown tree. Otte and Frazzoli [21] further address the problem of online planning in dynamic environments by modifying the original search graph whenever changes in the environment are observed. Among MAVs, Grzonka et al. [5] considered a quadrotor equipped with an on-board laser scanner, and scanned the environment, adapting the trajectory as new objects were spotted. Similarly, Nieuwenhuisen et al. [6] also used a 3D laser scanner on an autonomous quadrotor to build and update an obstacle map and replan collision-free trajectories. Similar approaches based on different sensors, such as cameras or depth sensors, were proposed in [22]. However, the previous approaches [5, 6, 22] rely on configurations that include other sensors (e.g, IMU, Laser Scanner) in addition to cameras. Furthermore, planning is performed without considering the visual perception and, in particular, the photometric information.

1.2 Contribution

In contrast to the previous works, in this paper we propose a novel method to update the optimal trajectory that leverages the *photometric information* (i.e, texture) and the 3D structure of newly-explored areas on the fly (i.e, *online*), avoiding full replanning. In order to use that information for minimizing pose uncertainty, we perform path planning in four degrees of freedom *(x, y, z, and yaw)*. Furthermore we proposed a novel *textured* volumetric map representation that allows us to efficiently synthesize views to compute the photometric information in the scene and plan accordingly. To the best of our knowledge, this is among the first works to propose a fully autonomous robotic system that performs onboard localization and online perception-aware planning. The main contributions of this paper are:

1. We propose to leverage the *photometric* appearance of the scene, in addition to the 3D structure, to select trajectories with minimum pose uncertainty. The photometric information is evaluated by using *direct* methods. As direct methods use all the information in the image, they provide a more robust and effective way to exploit visual information compared to feature-based strategies.
2. Perception-aware planning is performed online as the robot explores the surroundings, without prior knowledge of the full map of the environment.
3. A novel *textured* volumetric map formulation is proposed to efficiently synthesize views for perception-aware planning.
4. We demonstrate the effectiveness of our approach with experiments in both real-world and simulated environment with a MAV only equipped with vision sensors.

2 Perception-Aware Pose Uncertainty Propagation

The visual localization system relies on the availability of texture in the scene to reduce the pose estimation uncertainty. As a consequence, selecting the trajectory that is optimal with respect to the localization accuracy requires evaluation of the pose-uncertainty propagation along a candidate path and the uncertainty reduction associated with the photometric information in the scene.

2.1 Pose Propagation

We represent the pose of the robot as a 6 Degree of Freedom (DoF) transformation matrix $\mathbf{T}$, member of the *special Euclidean group* in $\mathbb{R}^3$, which is defined as follows:

$$\mathrm{SE}(3) := \left\{ \mathbf{T} = \begin{bmatrix} \mathbf{C} & \mathbf{r} \\ \mathbf{0}^T & 1 \end{bmatrix} \middle| \, \mathbf{C} \in \mathrm{SO}(3), \mathbf{r} \in \mathbb{R}^3 \right\}, \tag{1}$$

where SO(3) is the special orthogonal group in $\mathbb{R}^3$ (the set of spatial rotations, i.e, $\mathbf{C}\mathbf{C}^T = \mathbf{1}$, $\det \mathbf{C} = 1$) and $\mathbf{1}$ is the 3×3 identity matrix. The *Lie Algebra* associated to the SE(3) Lie Group is indicated as $\mathfrak{se}(3)$. To represent the uncertainty of the robot pose, we define a random variable for SE(3) members according to:

$$\mathbf{T} := \exp(\boldsymbol{\xi}^{\wedge})\bar{\mathbf{T}}, \tag{2}$$

where $\bar{\mathbf{T}}$ is a noise-free value that represents the pose and $\boldsymbol{\xi} \in \mathbb{R}^6$ is a small perturbation that we assume to be normally distributed $\mathcal{N}(\boldsymbol{\xi}|\mathbf{0}, \boldsymbol{\Sigma})$. We make use of the $\wedge$ operator to map $\boldsymbol{\xi}$ to a member of the Lie algebra $\mathfrak{se}(3)$ (see [23]).

We refer to $\mathbf{T}_{k,w}$ as the robot pose at time k relative to the world frame w and to $\mathbf{T}_{k+1,k}$ as the transformation between the pose at time k and $k+1$.

Assuming no correlation between the current pose and the transformation between k and $k+1$, we can represent $\mathbf{T}_{k,w}$ and $\mathbf{T}_{k+1,k}$ with their means and covariances $\{\bar{\mathbf{T}}_{k,w}, \boldsymbol{\Sigma}_{k,w}\}$ and $\{\bar{\mathbf{T}}_{k+1,k}, \boldsymbol{\Sigma}_{k+1,k}\}$, respectively. Combining them, we get

$$\mathbf{T}_{k+1,w} = \mathbf{T}_{k,w}\ \mathbf{T}_{k+1,k}. \tag{3}$$

To compute the mean and the covariance of the compound pose, we use the results from [23]. The mean and the covariance, approximated to fourth order, are:

$$\bar{\mathbf{T}}_{k+1,w} = \bar{\mathbf{T}}_{k,w}\ \bar{\mathbf{T}}_{k+1,k},\ \boldsymbol{\Sigma}_{k+1,w} \backsimeq \boldsymbol{\Sigma}_{k,w} + \mathcal{T}\boldsymbol{\Sigma}_{k+1,k}\mathcal{T}^{\top} + \mathcal{F} \tag{4}$$

where $\mathcal{T}$ is $Ad(\bar{\mathbf{T}}_{k,w})$, the adjoint operator for SE(3), and $\mathcal{F}$ encodes the fourth-order terms. Using Eq. (4), we propagate the uncertainty along a given trajectory.

2.2 *Measurement Update*

In contrast to previously published approaches, which mostly rely on *sparse* image features, we use direct methods, in the form of *dense image-to-model alignment*, for the measurement update. Integrating the intensity and depth of *every* pixel in the image enables us to consider photometric information when planning the trajectory.

2.2.1 Preliminary Notation

At each time step of the robot navigation, we can compute a dense surface model $\mathcal{S} \in \mathbb{R}^3 \times \mathbb{R}^+$ (3D position and grayscale intensity) of the explored part of the scene. The rendered synthetic image is denoted with $\mathbf{I}_s : \Omega_s \subset \mathbb{R}^2 \to \mathbb{R}^+$, where Ω_s is the image domain and $\mathbf{u} = (u, v)^T \in \Omega_s$ are pixel coordinates. Furthermore, we refer to the depthmap $\mathbf{D}_s$, associated to an image $\mathbf{I}_s$, as the matrix containing the distance at every pixel to the surface of the scene: $\mathbf{D}_s : \Omega_s \to \mathbb{R}^+; \mathbf{u} \mapsto d_{\mathbf{u}}$ where $d_{\mathbf{u}}$ is the depth associated to $\mathbf{u}$. A 3D point $\mathbf{p} = (x, y, z)^T$ in the camera reference frame is mapped to the corresponding pixel in the image $\mathbf{u}$ through the camera projection model $\pi : \mathbb{R}^3 \to \mathbb{R}^2$, $\mathbf{u} = \pi(\mathbf{p})$. On the other hand, we can recover the 3D point associated to the pixel $\mathbf{u}$ using the inverse projection function π^{-1} and the depth $d_{\mathbf{u}}$:

$$\mathbf{p_u} = \pi^{-1}(\mathbf{u}, d_\mathbf{u}). \tag{5}$$

Note that the projection function π is determined by the intrinsic camera parameters that are known from calibration. Finally, a rigid body transformation $\mathbf{T} \in \mathrm{SE}(3)$ rotates and translates a point $\mathbf{q}$ to:

$$\mathbf{q}'(\mathbf{T}) := (\mathbf{1} \,|\, \mathbf{0})\, \mathbf{T}\, (\mathbf{q}^T, 1)^T. \tag{6}$$

2.2.2 Dense Image-to-Model Alignment

To refine the current pose estimate, we use *dense image-to-model alignment* [18, 24] (see Fig. 2). This approach computes the pose $\mathbf{T}_{k,w}$ of the synthetic image $\mathbf{I}_s$ by minimizing the photometric error between the observed image and the synthetic one. Once converged, it also provides the uncertainty of the alignment by evaluating the *Fisher Information Matrix*, which we use to select informative trajectories.

The photometric error $r_\mathbf{u}$ for a pixel $\mathbf{u}$ is the difference of the intensity value at pixel $\mathbf{u}$ in the real image acquired at time step k and the intensity value in the synthetic image rendered at the estimated position $\hat{\mathbf{T}}_{k,w}$:

$$r_\mathbf{u} = \mathbf{I}_k(\mathbf{u}) - \mathbf{I}_s(\pi(\mathbf{p}'_\mathbf{u}(\hat{\mathbf{T}}_{k,w}))) \tag{7}$$

The error is assumed to be normally distributed $r_\mathbf{u} \sim \mathcal{N}(0, \sigma_i^2)$, where σ_i is the standard deviation of the image noise.

Due to the nonlinearity of the problem, we assume that we have an initial guess of the pose $\hat{\mathbf{T}}_{k,w}$ and iteratively compute update steps $\hat{\mathbf{T}}_{k,w} \leftarrow \exp(\boldsymbol{\xi}^\wedge)\hat{\mathbf{T}}_{k,w}$, $\boldsymbol{\xi}^\wedge \in \mathfrak{se}(3)$ that minimize the error. The update step minimizes the least-squares problem:

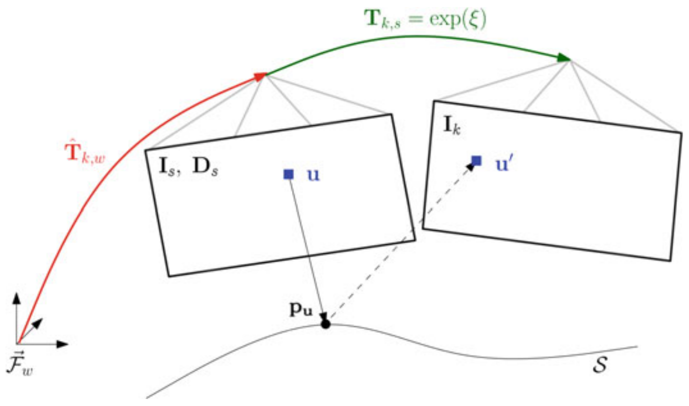


Fig. 2 Illustration of the dense image-to-model alignment used in the measurement update. Given an estimate of the pose $\hat{\mathbf{T}}_{k,w}$, we can synthesize an image and depthmap $\{\mathbf{I}_k, \mathbf{D}_k\}$ from the 3D model $\mathcal{S}$

$$\boldsymbol{\xi} = \arg\min_{\boldsymbol{\xi}} \sum_{\mathbf{u}\in\Omega_s} \frac{1}{2\sigma_i^2}\Big[\mathbf{I}_k(\mathbf{u}') - \mathbf{I}_s(\pi(\mathbf{p}'_{\mathbf{u}}(\hat{\mathbf{T}}_{k,w})))\Big]^2, \tag{8}$$

with $\mathbf{p_u}$ given by (5), $\mathbf{p}'_{\mathbf{u}}$ as in (6), and $\mathbf{u}' = \pi\big(\mathbf{p}'_{\mathbf{u}}(\exp(\boldsymbol{\xi}^\wedge))\big)$.

Addressing the least-squares problem (8) we can compute the optimal $\boldsymbol{\xi}$ using the Gauss-Newton method and solving the normal equations $\mathbf{J}^T\mathbf{J}\boldsymbol{\xi} = -\mathbf{J}^T\mathbf{r}$, where $\mathbf{J}$ and $\mathbf{r}$ are the stacked Jacobian and image residuals of all pixels $\mathbf{u} \in \Omega_s$, respectively.

At the convergence of the optimization, the quantity

$$\boldsymbol{\Lambda}_k = \frac{1}{\sigma_i^2}\mathbf{J}^T\mathbf{J} \tag{9}$$

is the *Fisher Information Matrix* and its inverse is the covariance matrix $\boldsymbol{\Sigma}_{\mathbf{I}_k}$ of the measurement update. According to [23], we find the covariance matrix after the measurement update at time k by computing

$$\boldsymbol{\Sigma}_{k,w} \leftarrow \left(\boldsymbol{\Lambda}_k^{-1} + \mathcal{J}^{-T}\boldsymbol{\Sigma}_{k,w}\mathcal{J}^{-1}\right)^{-1}, \tag{10}$$

where the "left-Jacobian" $\mathcal{J}$ is a function of how much the measurement update modified the estimate. Given the information matrix in (9), we define the photometric information gain as $\mathrm{tr}(\boldsymbol{\Lambda}_k)$.

3 Online Perception-Aware Path Planning

The framework described in Sect. 2 is able to predict the propagation of the pose uncertainty along a given trajectory by integrating the photometric information when available. However, to select the *best* sequence of camera viewpoints we need to evaluate all the possible trajectories. As we do not assume to have any given prior knowledge about the scene, the photometric information of the environment, as well as its 3D geometry, are unknown. Hence, the plan that is considered optimal in the beginning, will be adapted as new information is gathered by the robot.

In this section, we describe how we enhance the RRT* [10] with the perception-aware nature that takes benefit from the photometric information to select the trajectory that is optimal with respect to the localization accuracy.

The RRT* incrementally grows a tree in the state space by randomly sampling and connecting points through collision-free edges. Optimality is guaranteed through the *rewire* procedure, which checks for better connections when adding a new point to the tree. The tree is composed of a set of vertices V that represent points in the state space. Each vertex $v = \{x_v, \boldsymbol{\Sigma}_v, \boldsymbol{\Lambda}_v, c_v, p_v\} \in V$ is described via its state $x_v = \mathbf{T}_{v,w}$ (i.e, the pose relative to the vertex v with respect to the reference frame w), c_v being the accumulated cost of the trajectory up to v and a unique parent vertex p_v. In addition, we add the pose covariance $\boldsymbol{\Sigma}_v$ and the photometric information $\boldsymbol{\Lambda}_v$

(relative to the camera view associated to the pose x_v) to the vertex v in order to update the pose covariance according to the photometric information.

To select the best path among all possible trajectories $\mathcal{T}_i \in \mathcal{P}$, we minimize:

$$J(\mathcal{T}_i) = \sum_{j=1}^{N_i} \alpha \, \mathrm{Dist}(x_{v^i_j}, x_{v^i_{j-1}}) + (1-\alpha)\, \mathrm{tr}(x_{v^i_j}.\Sigma) \ , \tag{11}$$

Algorithm 1 Perception-aware RRT*

```
01: Init: $x_{v_0} = x_{\mathrm{init}}$; $p_{v_0} = \mathrm{root}$; $\boldsymbol{\Sigma}_{v_0} = \boldsymbol{\Sigma}_0$; $c_{v_0}$; $V = \{v_0\}$; Number of iterations $T$
02: for $t = 1, \dots, T$ do
03:   $x_{\mathrm{new}} = \mathrm{SampleUnexplored}()$
04:   $v_{\mathrm{nst}} = \mathrm{Nearest}(x_{\mathrm{new}})$
05:   if ObstacleFree($v_{\mathrm{new}}, v_{\mathrm{nst}}$)
06:     $\Sigma_{\mathrm{t}} = \mathrm{PropagateAndUpdate}(x_{v_{\mathrm{nst}}}, \boldsymbol{\Sigma}_{v_{\mathrm{nst}}}, x_{v_{\mathrm{new}}}, \boldsymbol{\Lambda}_{v_{\mathrm{new}}})$
07:     $J_{\min} = c_{v_{\mathrm{nst}}} + (1-\alpha)\,\mathrm{tr}(\Sigma_{\mathrm{t}}) + \alpha \mathrm{Dist}(x_{v_{\mathrm{nst}}}, x_{v_{\mathrm{new}}})$
08:     $v_{\min} = v_{\mathrm{nst}}$
09:     $V = V \cup v(x_{\mathrm{new}})$
10:     $V_{\mathrm{neighbors}} = \mathrm{Near}(V, v_{\mathrm{new}})$
11:     for all $v_{\mathrm{near}} \in V_{\mathrm{neighbors}}$ do
12:       if CollisionFree($v_{\mathrm{near}}, v_{\mathrm{new}}$)
13:         $\Sigma_{\mathrm{t}} = \mathrm{PropagateAndUpdate}(x_{v_{\mathrm{near}}}, \boldsymbol{\Sigma}_{v_{\mathrm{near}}}, x_{v_{\mathrm{new}}}, \boldsymbol{\Lambda}_{v_{\mathrm{new}}})$
14:         if $c_{v_{\mathrm{near}}} + (1-\alpha)\,\mathrm{tr}(\Sigma_{\mathrm{t}}) + \alpha \mathrm{Dist}(x_{v_{\mathrm{near}}}, x_{v_{\mathrm{new}}}) < J_{\min}$
15:           $J_{\min} = c_{v_{\mathrm{near}}} + (1-\alpha)\,\mathrm{tr}(\Sigma_{\mathrm{t}}) + \alpha \mathrm{Dist}(x_{v_{\mathrm{near}}}, x_{v_{\mathrm{new}}})$
16:           $\boldsymbol{\Sigma}_{v_{\mathrm{new}}} = \Sigma_{\mathrm{t}}$, $c_{v_{\mathrm{new}}} = J_{\min}$, $v_{\min} = v_{\mathrm{near}}$
17:         end if
18:       end if
19:       ConnectVertices($v_{\min}, v_{\mathrm{new}}$)
20:     end for
21:     RewireTree()
22:   end if
23: end for
```

where the trajectory is represented by a sequence of N_i waypoints v^i_j, $\mathrm{Dist}(\cdot,\cdot)$ computes the distance between two vertices and α defines the trade-off between this distance and the photometric information gain. Note that we jointly optimize for position and yaw orientation w.r.t. information gain, so the optimal poses are not just the RRT* poses with optimized orientation. We choose the trace to include the visual information into the cost function following the considerations in [25]. In particular, the minimization of the trace of the pose covariance matrix (A-optimality) guarantees that the majority of the state space dimensions are be considered (in contrast to the D-optimality), but does not require us to compute all the eigenvalues (E-optimality). The fundamental steps of the perception-aware RRT* are summarized in Algorithm 1. At each iteration, it samples a new state from the state space and connects it to the nearest vertex (lines 3–19). Next, the function `Near()` checks on the vertices within a ball, centered at the sampled state (see [10]), and propagate the pose covariance from these vertices to the newly sampled one. The one that minimizes the cost function (11) gets selected. Finally, we update the tree connections through the *rewire* procedure. Note that although the optimization is performed on the trace, the full covariance is propagated along each trajectory for evaluation.

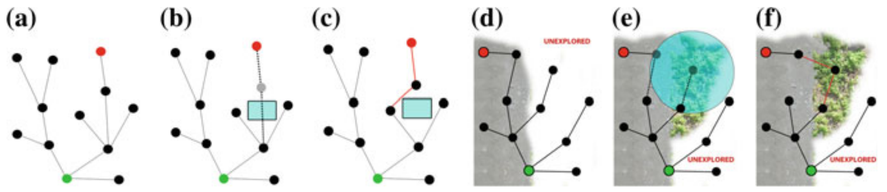


Fig. 3 Online update steps during exploration: Figures **a**–**c** depict the subtree invalidation and rewiring update when an obstacle is spotted, while **d**–**f** show how the tree is rewired when new photometric information is available from the scene

Given an initially optimal path, we can now start exploring the environment. When new parts of the scene are revealed, the current trajectory might become non-optimal or even infeasible in case of obstacles. One possibility would be to recompute the tree from scratch after every map update but this would be costly and computationally intractable to have the system integrated into an MAV application. For this reason, we propose to update the planning tree *on-the-fly* by only processing vertices and edges affected by new information. This online update is illustrated in Fig. 3 and its fundamental steps are depicted in Algorithm 2. Consider an initial planning tree as in Fig. 3a, that is grown from a starting point (indicated by a green circle) to a desired end point location (the red circle). Whenever a new obstacle is spotted, the respective edge and the affected subtree get invalidated and regrown (lines 04–06) as in Fig. 3b. Note that the `SampleUnexplored()` function is now bounded within the subspace corresponding to the invalidated subtree, which results in a drastically reduced number of iterations compared to fully regrowing the RRT* tree from scratch. The second scenario in Fig. 3d–f demonstrates the case of gaining areas with distinctive photometric information. As newly discovered areas provide photometric information, as shown in Fig. 3e, the neighboring vertices are updated by the `RewireTree()` procedure (lines 07–10 in Algorithm 2). Potentially better connections are considered to form a new path with lower costs (Fig. 3f).

Algorithm 2 Online perception-aware RRT*

```
01: while 1 do
02:   UpdateCollisionMap()
03:   UpdatePhotometricInformationMap()
04:   $V_{\text{colliding}}$ = NewCollidingVertices()
05:   InvalidateSubTree($V_{\text{colliding}}$)
06:   Run PerceptionAwareRRT* 1
07:   $V_{\text{inf}}$ = UpdatedVertices()
08:   for all $v_{\text{inf}} \in V_{\text{inf}}$ do
09:     $\Lambda_v = \Lambda_v^{new}$
10:     RewireTree()
11:   end for
12: end while
```

4 Textured Volumetric Mapping

We implement an extension to the popular *OctoMap* [26] 3D mapping framework that records texture information within the volumetric map, allowing novel views to be synthesized for perception-aware path planning (see Fig. 4).

We utilize this stored texture to synthesize views of the known map from hypothetical positions for the camera. For each synthetic view, we synthesize an image of what it would look like to observe the environment from that pose—at least up to the currently observed state of the map—and use these synthetic images in the computation of information gain during planning. As an extension to an OctoMap that stores an estimate of occupancy probability for each voxel, we maintain an estimate of the texture for *each face* of each voxel as an intensity value, averaged over all of the observations of that face. We chose this approach because of its compactness—we must only store the current estimate and the number of cumulative observations—and because it is not depth dependent for either updating or querying. It is also directly extensible to a hierarchical representation, such that texture values at higher levels of the octree can be computed from the faces of their child voxels. While our approach to rendering images from a volumetric map is similar to the one in [27], we chose to store texture for the faces, and not just for the volume, because the space represented by a voxel does not necessarily have the same appearance when observed from different sides. Storing more descriptive representations of texture (e.g. Harris corner scores) for the faces would be beneficial, but these metrics are often dependent on the range at which they are observed, presenting a barrier for maintaining a general estimate. The average intensity representation is efficient to update with new observations, efficient to query for the current estimate, and adds only minimal overhead to the computation required for mapping.

Our update method proceeds as follows. Given an input point cloud, occupancy is updated as in [26], where ray casting from the sensor origin is used to update each leaf voxel along the ray, until the observed point, and each leaf voxel is updated at most once for occupancy. To update the texture, for each point p_i^k in the k^{th} point cloud C^k, we determine the face f that its ray intersects in the leaf voxel n containing p_i^k. At leaf voxel n, we maintain the current intensity estimate t_f and number of cumulative observations m_f for each face $f \in 1 \ldots 6$ of the voxel cube (see Fig. 4a). After the insertion of k point clouds, these quantities are:

$$m_f^k = \sum_{j=1}^{k} \left| p_i^j \in n \right|, \qquad t_f^k = \frac{\sum_{j=1}^{k} t_{p_i^j} \in f}{m_f} \tag{12}$$

This can be updated efficiently for each new point cloud input:

$$m_f^{k+1} = m_f^k + \left| p_i^{k+1} \in n \right|, \qquad t_f^{k+1} = \frac{m_f^k t_f^k + t_{p_i^k}}{m_f^{k+1}} \tag{13}$$

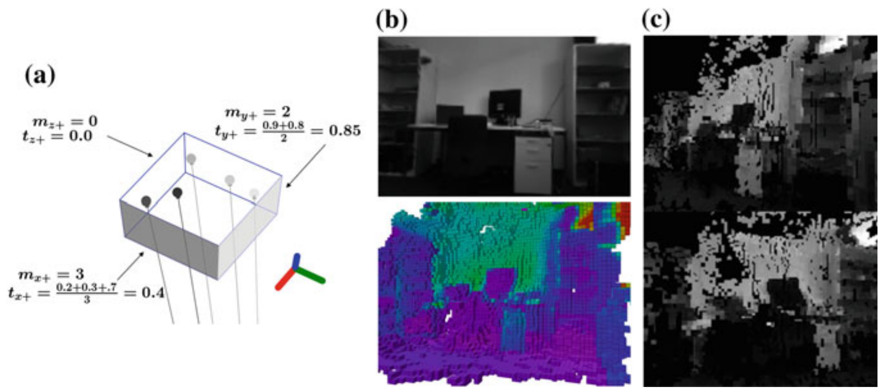

(a)
(b)
(c)
$m_{z+} = 0$
$t_{z+} = 0.0$
$m_{y+} = 2$
$t_{y+} = \frac{0.9+0.8}{2} = 0.85$
$m_{x+} = 3$
$t_{x+} = \frac{0.2+0.3+.7}{3} = 0.4$

Fig. 4 Textured volumetric mapping: texture information is stored for each face of each voxel in the OctoMap. Each face maintains a mean intensity value for all of the sensor observations that have intersected with it when adding data to the map (**a**). A visualization of a *Textured Octomap* is shown in **b**, where an office scene was observed with a handheld stereo camera. In **c**, we have synthesized some images from the map, at poses that the camera has not yet observed

The inclusion of texture in the OctoMap requires an additional computational overhead of only 15% for both insertion and querying.

Storing texture in a volumetric map allows us to hypothesize about the photometric information that our robot could obtain if it moved to a particular pose. We do this by synthesizing images of the map from a hypothetical pose, casting rays through each pixel in the image into the map (See Fig. 4c). When these rays intersect with the face of an occupied voxel, we record the texture of the face and the depth to that voxel in intensity and depth images. These synthetic images are generated for each sampled pose when the planner generates or rewires the tree.

5 System Overview

We consider an MAV that explores an unknown environment by relying only on its camera to perform localization, dense scene reconstruction and optimal trajectory planning. We have integrated the online perception-aware planner with two different mapping systems (see Fig. 5): a monocular dense reconstruction system that generates a point cloud map, and a volumetric system that uses stereo camera input.

In the monocular system, the localization of the quadrotor runs onboard, providing the egomotion estimation to perform navigation and stabilization. To achieve real-time performance, the dense map reconstruction and the online perception-aware path planning runs off-board on an Intel i7 laptop with a GPU, in real-time.

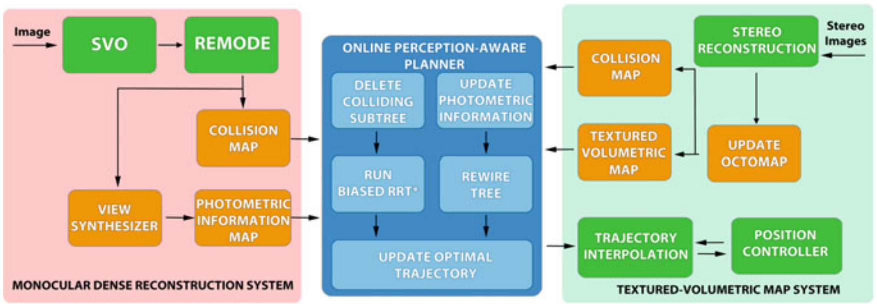


Fig. 5 Block diagram of the online perception-aware planning system

At each time step k, the quadrotor receives a new image to perform egomotion estimation. We use the Semi-direct monocular Visual Odometry (SVO) proposed in [2], which allows us to estimate the quadrotor motion in real-time. The computed pose $\mathbf{T}_{k,w}$ and the relative image are then fed into the dense map reconstruction module (REMODE [28], a probabilistic, pixelwise depth estimate to compute dense depthmaps). Afterwards, the dense map provided by the reconstruction module is sent to the path planning pipeline and is used to update both the collision map (using Octomap [26]) and the photometric information map. The last one is then used to update $\boldsymbol{\Lambda}_v$ for each vertex affected by the map update. Finally, we update the optimal trajectory following the procedure described in Algorithm 2.

For the textured volumetric map system, we take input from a stereo camera, perform egomotion estimation with SVO as above, and compute a dense depth map with OpenCV's Block Matcher. The estimated camera pose from SVO and the point cloud produced from the depth map are used to update the Textured OctoMap as in Sect. 4. This volumetric map serves as a collision map, when it is queried for occupancy, and is used to synthesize views and compute photometric information gain during planning, when it is queried for texture. This pipeline runs in real time onboard an MAV's embedded single board computer (an Odroid XU3 Lite) using a map with 5 cm resolution, and with the input images downsampled by a factor of 4 to 188×120, and throttled down to 1 Hz. However, we evaluate this system in simulation, and for the experiments in Sect. 6.2, we run the simulation, visual pipeline, planner, and control software all on a laptop with an Intel i7.

6 Experiments

6.1 Real World Experiments

We motivate our approach by discussing how the photometric information distribution changes over time when exploring an unknown environment. Figure 6 shows the

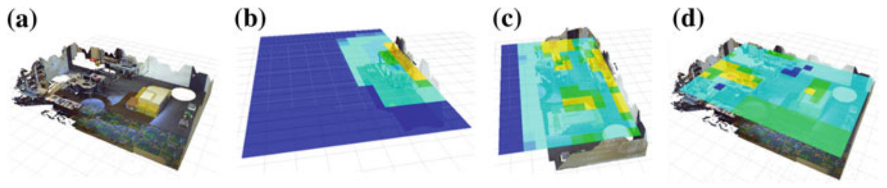


Fig. 6 Computed photometric information gain at different exploration stages (**b**, **c** and **d**) for the scene in **a**. Warm (*yellowish*) colors refer to camera viewpoints exhibiting a higher amount of texture, while the cool (*bluish*) ones indicate less informative areas

map for the photometric information gain at different exploration stages. In Fig. 6b the almost unexplored scene has very little valuable information to compute a reliable plan. Standard planners, which calculate trajectories only once without performing online updates, compute sub-optimal plans or even collide with undiscovered objects. Hence, an online approach is needed to re-plan as new photometric information is gathered (see Fig. 6c, d).

To evaluate the proposed online perception-aware path planning, we ran experiments on an indoor environment with different configurations. We set up two scenarios with different object arrangements to vary the texture and the 3D structure of the scene. In the first scenario, the monocular camera on the MAV is downward-looking, while in the last one we choose a front-looking monocular configuration with an angle of 45 degrees with respect to the ground plane. We made experiments with two different camera setups to investigate the influence of the camera viewpoint on the optimal trajectory computation. Intuitively, the front-looking configuration provides more information since also areas far from the quadrotor are observed. Conversely, with the downward-looking configuration, the pose estimation algorithm is more reliable, but less information is captured from the scene. Finally, in all the experiments we set $\alpha = 0.1$ to increase the importance of the pose uncertainty minimization.

In all the scenarios, we put highly-textured carpets and boxes along the walls, while the floor in the center of the room is left without texture (i.e, with a uniform color). In the first scenario, we also put an obstacle in the center of the room. At the beginning of the exploration, the planner shows similar behavior in all the experiments (see Fig. 7a, d). The information about the scene is very low, thus, our approach computes a simple straight trajectory to the goal. As the robot explores the environment, the plan is updated by preferring areas with high photometric information (cf. Fig. 7b, c). In the second scenario, a front-looking camera provides photometric information about areas distant from the current MAV pose. As a consequence, we obtain an optimal plan earlier with respect to the first experiment (see Fig. 7e, f).

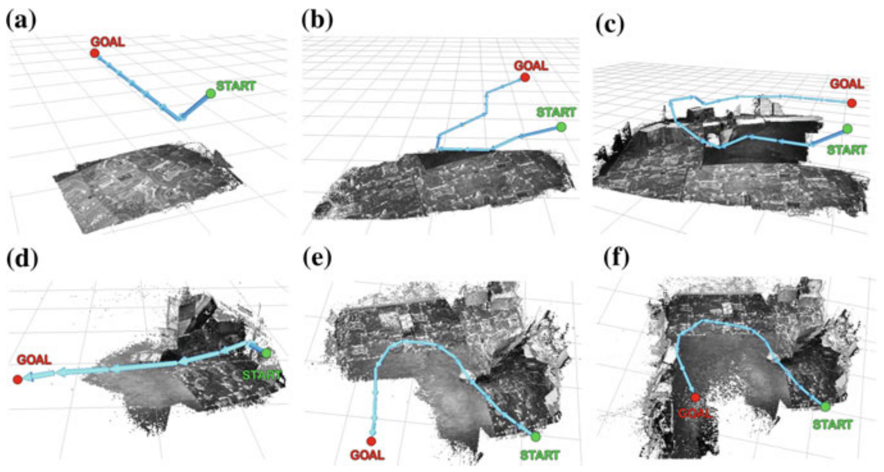


Fig. 7 Experimental results in two real scenarios (rows). The first column shows the initially computed trajectories, only having little information of the environment available. The second and third column demonstrate the update of the plan as new information is gathered by updating the scene

6.2 Simulated Experiments

We demonstrate the proposed system in a simulated environment, using the components described in Sect. 5. Two trials were performed in environments simulated with *Gazebo*, one designed to explicitly test perception (*labyrinth*) and one designed to simulate a real world environment (*kitchen*). The labyrinth scenario is designed with flat and highly-textured walls to test the capability of our perception-aware planner to choose the MAV orientations that maximize the amount of photometric information. The quadrotor starts in one of the two long corridors in the scene (see Fig. 8a) and is asked to reach the goal location that is located at 25 m from the start location. In the kitchen world (see Fig. 8d), the MAV begins at a position that is separated by two walls from the goal location, which is 12.5 m away. We compare the performance of the standard RRT* planner and our perception-aware planner in Figs. 8 and 9.

6.3 Discussion

The qualitative results shown for the real world (Fig. 7) and simulated (Fig. 8) experiments show that the perception aware planner does indeed choose trajectories that allow the MAV to observe more photometric information. Quantitatively, this results in a dramatic improvement in the uncertainty of the vehicle's pose estimate. The results in Fig. 9 show that the pose uncertainty, measured as the trace of the covari-

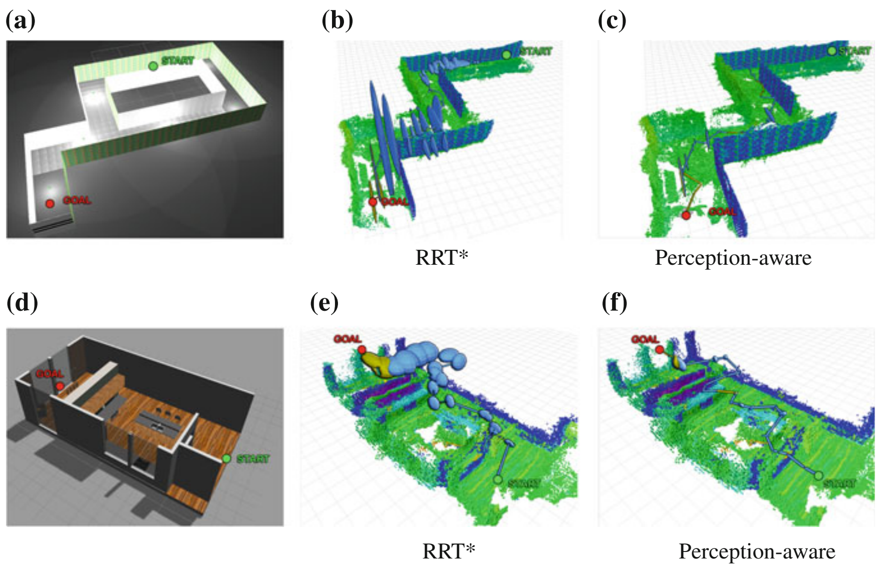


Fig. 8 Exploration trial in the *labyrinth* (**a**) and in the *kitchen* (**d**) simulated environments. The trajectories computed by the RRT* planner are shown in **b** for the labyrinth scenario and in **e** for the kitchen, while the ones computed with the perception aware planner are shown in **c** and **f**, respectively. The Textured OctoMaps are visualized with a color corresponding to the mean intensity over all of the observed faces, with *red* representing high intensity, and *purple* representing low intensity. The pose covariance at each waypoint is shown as an ellipse, with the most recent update in *orange*, and the rest of the plan in *blue*

ance matrix and visualized as ellipses in Fig. 8, is up to an order of magnitude smaller when the planner considers the texture of the environment.

In both of the simulated experiments, the RRT* and perception aware planners both reached the goal location in all trials. On average, for the *labyrinth* it took 718.3 and 715.2 s, respectively, and for *kitchen* it took 578.3 and 580.4 s, respectively. The results are shown in Fig. 8b, c for the labyrinth tests and in Fig. 8e, f for the kitchen ones. The most important distinction in this performance comparison is the pose uncertainty across the trajectory. The two planners produce similar trajectories in terms of waypoint positions, but the covariances for the RRT* trajectory are much larger due to the desired yaw angles that are chosen for the waypoints. The proposed perception aware planner specifically optimizes the waypoint position and yaw angle (i.e. *where to look*) in order to minimize this pose uncertainty. As a consequence, the plan computed with our strategy has low pose uncertainty values, while the RRT* trajectory, which does not consider the visual information, leads to very low localization accuracy, which can make the navigation infeasible due to the high risk of collisions.

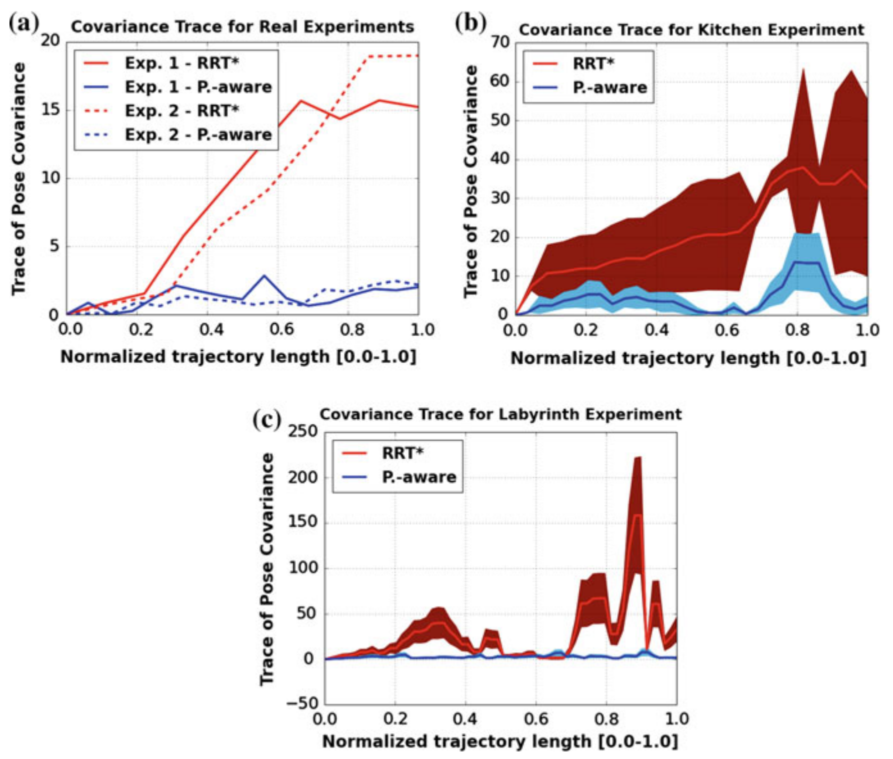


Fig. 9 Quantitative results for our experiments showing the evolution of the MAV's pose covariance during the planned trajectory. **a** shows results of the real world experiments. **b** and **c** show the simulated *kitchen* and *labyrinth* trials, respectively. The plans for each trial result in different length trajectories, so the length of each trajectory is normalized to one. For each simulated experiment, we conducted 15 trials, normalized the trajectories, and inferred Gaussian distributions at each point in a set of equally-spaced samples along a normalized trajectory. In **b** and **c**, each *solid line* represents the mean over all of the trials, and the *colored* band is the 95% confidence interval

7 Conclusions

We have proposed a novel approach for performing online path planning that leverages the photometric information in the environment to plan a path that minimizes the pose uncertainty of a camera-equipped MAV that is performing vision-based egomotion estimation. These advances include a perception-aware path planner and a textured volumetric map. This planning framework has been evaluated with real and simulated experiments, and both the qualitative and quantitative results support the conclusion that taking photometric information into account when planning significantly reduces a vision-controlled MAV's pose uncertainty. Utilizing perception awareness will enable more robust vision-controlled flight in arbitrary environments.

References

1. Engel, J., Schöps, T., Cremers, D.: LSD-SLAM: large-scale direct monocular SLAM. In: European Conference on Computer Vision (ECCV), pp. 834–849 (2014)
2. Forster, C., Pizzoli, M., Scaramuzza, M.: SVO: fast semi-direct monocular visual odometry. In: IEEE International Conference on Robotics and Automation (ICRA) (2014)
3. Weiss, S., Achtelik, M.W., Lynen, S., Achtelik, M.C., Kneip, L., Chli, M., Siegwart, R.: Monocular vision for long-term micro aerial vehicle state estimation: a compendium. J. Field Robot. **30**(5), 803–831 (2013)
4. Scaramuzza, D., Achtelik, M.C., Doitsidis, L., Fraundorfer, F., Kosmatopoulos, E.B., Martinelli, A., Achtelik, M.W., Chli, M., Chatzichristofis, S.A., Kneip, L., Gurdan, D., Heng, L., Lee, G.H., Lynen, S., Meier, L., Pollefeys, M., Renzaglia, A., Siegwart, R., Stumpf, J.C., Tanskanen, P., Troiani, C., Weiss, S.: Vision-controlled micro flying robots: from system design to autonomous navigation and mapping in GPS-denied environments. IEEE Robot. Autom. Mag. **21**, 26–40 (2014)
5. Grzonka, S., Grisetti, G., Burgard, W.: A fully autonomous indoor quadrotor. IEEE Trans. Robot. **28**(1), 90–100 (2012)
6. Nieuwenhuisen, M., Droeschel, D., Beul, M., Behnke, S.: Obstacle detection and navigation planning for autonomous micro aerial vehicles. In: 2014 International Conference on Unmanned Aircraft Systems (ICUAS), pp. 1040–1047. IEEE (2014)
7. Bry, A., Roy, N.: Rapidly-exploring random belief trees for motion planning under uncertainty. In: IEEE International Conference on Robotics and Automation (ICRA), pp. 723–730 (2011)
8. Bachrach, A., Prentice, S., He, R., Henry, P., Huang, A.S., Krainin, M., Maturana, D., Fox, D., Roy, N.: Estimation, planning, and mapping for autonomous flight using an RGB-D camera in GPS-denied environments. Int. J. Robot. Res. **31**(11), 1320–1343 (2012)
9. Sim, R., Roy, N.: Global a-optimal robot exploration in slam. In: IEEE International Conference on Robotics and Automation (ICRA), pp. 661–666, April 2005
10. Karaman, S., Frazzoli, E.: Sampling-based algorithms for optimal motion planning. Int. J. Robot. Res. **30**(7), 846–894 (2011)
11. Chen, S., Li, Y., Kwok, N.M.: Active vision in robotic systems: a survey of recent developments. Int. J. Robot. Res. **30**(11), 1343–1377 (2011)
12. Soatto, S.: Actionable information in vision. In: Cipolla, R., Battiato, S., Farinella, G.M. (eds.) Machine Learning for Computer Vision, pp. 17–48. Springer, Heidelberg (2013)
13. Davison, A.J., Murray, D.W.: Simultaneous localization and map-building using active vision. IEEE Trans. Pattern Anal. Mach. Intell. **24**(7), 865–880 (2002)
14. Sadat, S.A., Chutskoff, K., Jungic, D., Wawerla, J., Vaughan, R.: Feature-rich path planning for robust navigation of MAVs with mono-SLAM. In: IEEE International Conference on Robotics and Automation (ICRA) (2014)
15. Achtelik, M.W., Lynen, S., Weiss, S., Chli, M., Siegwart, R.: Motion-and uncertainty-aware path planning for micro aerial vehicles. J. Field Robot. **31**(4), 676–698 (2014)
16. Kim, A., Eustice, R.M.: Perception-driven navigation: active visual slam for robotic area coverage. In: IEEE International Conference on Robotics and Automation (ICRA), pp. 3196–3203. IEEE (2013)
17. Irani, M., Anandan, P.: All about direct methods. In: Vision Algorithms: Theory and Practice, pp. 267–277. Springer (2000)
18. Newcombe, R.A., Lovegrove, S.J., Davison, A.J.: DTAM: dense tracking and mapping in real-time. In: IEEE International Conference on Computer Vision (ICCV), pp. 2320–2327 (2011)
19. Ferguson, D., Kalra, N., Stentz, A.: Replanning with rrts. In: IEEE International Conference on Robotics and Automation (ICRA), pp. 1243–1248. IEEE (2006)
20. Boardman, B.L., Harden, T.A., Martinez, S.: Optimal kinodynamic motion planning in environments with unexpected obstacles. Technical report, Los Alamos National Laboratory (LANL) (2014)

21. Otte, M., Frazzoli, E.: RRT-X: real-time motion planning/replanning for environments with unpredictable obstacles. In: International Workshop on the Algorithmic Foundations of Robotics (WAFR) (2014)
22. Schmid, K., Lutz, P., Tomić, T., Mair, E., Hirschmüller, H.: Autonomous vision-based micro air vehicle for indoor and outdoor navigation. J. Field Robot. **31**(4), 537–570 (2014)
23. Barfoot, T.D., Furgale, P.T.: Associating uncertainty with three-dimensional poses for use in estimation problems. IEEE Trans. Robot. **30**, 679–693 (2014)
24. Meilland, M., Comport, A.I.: On unifying key-frame and voxel-based dense visual SLAM at large scales. In: IEEE International Conference on Intelligent Robots and Systems (IROS) (2013)
25. Haner, S., Heyden, A.: Optimal view path planning for visual slam. In: Heyden, A., Kahl, F. (eds.) Image Analysis, pp. 370–380. Springer, Heidelberg (2011)
26. Hornung, A., Wurm, K.M., Bennewitz, M., Stachniss, C., Burgard, W.: Octomap: an efficient probabilistic 3d mapping framework based on octrees. Auton. Robot. **34**(3), 189–206 (2013)
27. Mason, J., Ricco, S., Parr, R.: Textured occupancy grids for monocular localization without features. In: IEEE International Conference on Robotics and Automation (ICRA), Shanghai, China (2011)
28. Pizzoli, M., Forster, C., Scaramuzza, D.: REMODE: probabilistic, monocular dense reconstruction in real time. In: IEEE International Conference on Robotics and Automation (ICRA), pp. 2609–2616. IEEE, May 2014

Optimal-State-Constraint EKF for Visual-Inertial Navigation

Guoquan Huang, Kevin Eckenhoff and John Leonard

1 Introduction

It is prerequisite for a robot (mobile sensing platform) to be able to determine its position and orientation (pose) in 3D for a variety of applications ranging from exploration of uncharted territory to augment reality for gaming. Over the past decades, an inertial navigation system (INS) using an inertial measurement unit (IMU) is among the most popular approaches to estimate the 6 degrees-of-freedom (d.o.f.) robot's poses, especially in GPS-denied environments such as underwater, indoor, in urban canyon, and on other planets. While simple integration of IMU measurements that are corrupted by noise and bias, often results in estimates unreliable in a long term, a camera that is small, light-weight, and energy-efficient, provides rich information about the perceived environment and serves as an idea aiding source for INS, i.e., visual-inertial navigation system (VINS). This problem is challenging in part because of the lack of global information to reduce the motion drift accumulated over time, which is exacerbated if low-cost sensors are used.

To date, various algorithms are available for VINS problems including visual-inertial SLAM [16] and visual-inertial odometry (VIO) [17], such as the extended Kalman filter (EKF) [16, 23], the unscented Kalman filter (UKF) [3], and the batch or incremental smoothers [13, 28], among which the EKF-based approach remains arguably the most popular because of its efficiency. For example, as a

G. Huang (✉) · K. Eckenhoff
Department of Mechanical Engineering, University of Delaware,
Newark, DE 19716, USA
e-mail: ghuang@udel.edu

K. Eckenhoff
e-mail: keck@udel.edu

J. Leonard
Computer Science and Artificial Intelligence Laboratory,
Massachusetts Institute of Technology, Cambridge, MA 02139, USA
e-mail: jleonard@mit.edu

A. Bicchi and W. Burgard (eds.), *Robotics Research*, Springer Proceedings in Advanced Robotics 2, DOI 10.1007/978-3-319-51532-8_8

state-of-the-art solution of VINS on mobile devices, Google's Project Tango [4] uses the EKF to tightly fuse the visual and inertial measurements by processing them in a single estimation thread. Although it works well in a relatively small-scale environment over a short period of time, yet it is not robust enough for widespread long-term deployments in critical applications such as search and rescue. This is due to the fact that any engineered sensor system has only finite resources available (e.g., processing power and memory storage). If sensor measurements are not optimally utilized, for example, due to stringent resources, the standard EKF can easily become vulnerable to errors and produce inconsistent estimates [6, 7, 10, 17].

In this paper, we develop a novel optimal-state-constraint (OSC)-EKF algorithm which essentially introduces a new methodology of efficiently processing visual information during VINS. As in [22], the proposed OSC-EKF performs tightly-coupled visual-inertial sensor fusion over a sliding window that consists of the current navigation state and the past cloned poses only (i.e., without including features in the state vector). This results in the complexity independent of the size of the environment (the total number of the features available in the environment, say M), while being dependent on the number of features observed in the current sliding window which is typically much smaller than M. The key idea of the proposed approach is to design a measurement model that utilizes all the feature measurements available within the current sliding window to impose probabilistically *optimal* constraints among the poses, while without estimating these features as part of the state vector. To that end, during each sliding window, we perform structure and motion using only the available camera measurements (which can be solved efficiently by leveraging many open-source optimized solvers), and subsequently marginalize out the structure (features). This results in the optimal motion constraints that will be used by the EKF to update the state estimates. The proposed approach is preliminarily validated in real-world experiments. Note that this methodology can also be generalized to other sensing modalities such as lidar and acoustic sensors.

The remainder of the paper is organized as follows: After an overview of related work in the next section, the standard EKF-based VINS is briefly described in Sect. 3. In Sect. 4, we present in detail the proposed OSC-EKF algorithm, which derives an optimal-state-constraint measurement model by marginalizing out the structure from the structure-and-motion process. In Sect. 5, the proposed OSC-EKF is validated in real-world experiments. Finally, Sect. 6 outlines the main conclusions of this work, as well as possible future research directions.

2 Related Work

Fusing visual and inertial measurements provides a popular means for robots navigating in 3D, in part because of the complementary sensing modalities and the reduced cost and size of the sensors as well as the increased computing power (e.g., see [2, 3, 6, 7, 11, 13–17, 19, 23, 26, 28, 31, 32] and references therein). Such visual-inertial navigation can be broadly categorized into loosely- and tightly-coupled approaches.

The former processes the IMU measurements and images separately in the front end to generate motion constraints, which subsequently are fused in the back end (e.g., [13, 32]). However, although this type of methods have advantage of computational efficiency, the decoupling results in information loss [17]. The latter seamlessly fuses the visual and inertial measurements by processing them in a single estimation thread (e.g., [6, 7, 10, 14, 15, 17]). As mentioned earlier, the proposed OSC-EKF also falls into the latter category.

As system observability plays an important role in developing VINS algorithms, the VINS observability/identifiability was studied by examining the system's indistinguishable trajectories [5, 14]. Similarly, Martinelli [19] employed the concept of continuous symmetries to show that in VINS, the IMU biases, 3D velocity, and absolute roll and pitch angles are observable. Moreover, in analogy to robot localization in 2D [8, 9], consistency of EKF-based VINS has been investigated from the perspective of observability [6, 7, 17]. In particular, Li and Mourikis [17] studied the impact of filter inconsistency due to the VINS observability properties, and leveraged the first-estimates-Jacobian methodology [9] to mitigate the inconsistency. Hesch et al. [6, 7] employed the observability-based methodology proposed in our prior work [8] and introduced the observability-constrained VINS, which ensures correct observability of the EKF linearized system. In our recent work [10], besides imposing the same observability constraints, we further enforce the propagation Jacobian to satisfy the semigroup property and thus to be a valid state transition matrix, which results in an alternative way of computing propagation Jacobians. Note that the same observability-based constraints may be used in the proposed approach to improve VINS consistency, which, however, is not the focus of this work. Instead, we here primarily focus on deriving a new optimal-state-constraint measurement model to better utilize measurement information, which can be used for any VINS formulation (see Sect. 4).

Recently the incremental light bundle adjustment (iLBA) [11, 12] extracts motion constraints from camera's feature measurements based on *three-view* geometry in the front end (without including features in the state vector as well), while in the back end, performing incremental smoothing for factor-graph based formulation of robot navigation. However, the proposed approach can utilize any number of views ($N \geq 2$) and derive optimal ego-motion constraints (up to scale) about the camera poses within the current sliding window and thus can be considered as generalized *N-view* constraints.

3 Visual-Inertial Navigation

In this section, we describe the IMU propagation and camera measurement models within the standard EKF framework, which govern the VINS. In order to lay the groundwork for the proposed approach, we here consider the SLAM scenario where only a single feature is included in the state vector, while the results can be readily extended to the case of multiple features.

3.1 IMU Propagation Model

The EKF uses the IMU (gyroscope and accelerometer) measurements for state propagation, and the state vector consists of the IMU states $\mathbf{x}_I$ and the feature position ${}^G\mathbf{p}_f$[1]:

$$\mathbf{x} = \begin{bmatrix}\mathbf{x}_I^T & {}^G\mathbf{p}_f^T\end{bmatrix}^T = \begin{bmatrix}{}^I_G\bar{\mathbf{q}}^T & \mathbf{b}_g^T & {}^G\mathbf{v}^T & \mathbf{b}_a^T & {}^G\mathbf{p}^T & {}^G\mathbf{p}_f^T\end{bmatrix}^T \tag{1}$$

where ${}^I_G\bar{\mathbf{q}}$ is the unit quaternion that represents the rotation from the global frame of reference $\{G\}$ to the IMU frame $\{I\}$ (i.e., different parametrization of the rotation matrix $\mathbf{C}({}^I_G\bar{\mathbf{q}})$); ${}^G\mathbf{p}$ and ${}^G\mathbf{v}$ are the IMU position and velocity in the global frame; and $\mathbf{b}_g$ and $\mathbf{b}_a$ denote the gyroscope and accelerometer biases, respectively.

By noting that the feature is static (with trivial dynamics), as well as using the IMU motion dynamics [29], the continuous-time dynamic equations of the state (1) are given by:

$$\begin{aligned}
&{}^I_G\dot{\bar{\mathbf{q}}}(t) = \frac{1}{2}\boldsymbol{\Omega}\left({}^I\boldsymbol{\omega}(t)\right){}^I_G\bar{\mathbf{q}}(t)\ ,\ \ {}^G\dot{\mathbf{p}}(t) = {}^G\mathbf{v}(t)\ ,\ \ {}^G\dot{\mathbf{v}}(t) = {}^G\mathbf{a}(t)\\
&\dot{\mathbf{b}}_g(t) = \mathbf{n}_{wg}(t)\ ,\ \ \dot{\mathbf{b}}_a(t) = \mathbf{n}_{wa}(t)\ ,\quad {}^G\dot{\mathbf{p}}_f(t) = \mathbf{0}_{3\times 1}
\end{aligned} \tag{2}$$

where ${}^I\boldsymbol{\omega} = \begin{bmatrix}\omega_1 & \omega_2 & \omega_3\end{bmatrix}^T$ is the rotational velocity of the IMU, expressed in $\{I\}$, ${}^G\mathbf{a}$ is the IMU acceleration in $\{G\}$, $\mathbf{n}_{wg}$ and $\mathbf{n}_{wa}$ are the white Gaussian noise processes that drive the IMU biases, and $\boldsymbol{\Omega}(\boldsymbol{\omega})$ is defined by:

$$\boldsymbol{\Omega}(\boldsymbol{\omega}) = \begin{bmatrix}-\lfloor\boldsymbol{\omega}\times\rfloor & \boldsymbol{\omega}\\ -\boldsymbol{\omega}^T & 0\end{bmatrix}\ ,\quad \lfloor\boldsymbol{\omega}\times\rfloor = \begin{bmatrix}0 & -\omega_3 & \omega_2\\ \omega_3 & 0 & -\omega_1\\ -\omega_2 & \omega_1 & 0\end{bmatrix}$$

A typical IMU provides gyroscope and accelerometer measurements, $\boldsymbol{\omega}_m$ and $\mathbf{a}_m$, both of which are expressed in the IMU local frame $\{I\}$:

$$\boldsymbol{\omega}_m(t) = {}^I\boldsymbol{\omega}(t) + \mathbf{b}_g(t) + \mathbf{n}_g(t) \tag{3}$$

$$\mathbf{a}_m(t) = \mathbf{C}({}^I_G\bar{\mathbf{q}}(t))\left({}^G\mathbf{a}(t) - {}^G\mathbf{g}\right) + \mathbf{b}_a(t) + \mathbf{n}_a(t) \tag{4}$$

where ${}^G\mathbf{g}$ is the gravitational acceleration expressed in $\{G\}$, and $\mathbf{n}_g$ and $\mathbf{n}_a$ are zero-mean, white Gaussian noise.

[1]Throughout this paper the subscript $\ell|j$ refers to the estimate of a quantity at time-step ℓ, after all measurements up to time-step j have been processed. $\hat{x}$ is used to denote the estimate of a random variable x, while $\tilde{x} = x - \hat{x}$ is the error in this estimate. $\mathbf{I}_n$ and $\mathbf{0}_n$ are the $n \times n$ identity and zero matrices, respectively. $\mathbf{e}_1, \mathbf{e}_2$ and $\mathbf{e}_3 \in \mathbb{R}^3$ are the unit vectors along $x-$, $y-$ and $z-$axes. Finally, the left superscript denotes the frame of reference which the vector is expressed with respect to.

Linearization of (2) at the current state estimates yields the continuous-time state-estimate propagation model [23]:

$$
\begin{aligned}
&{}^I_G\dot{\hat{\bar{\mathbf{q}}}}(t) = \frac{1}{2}\boldsymbol{\Omega}\left({}^I\hat{\boldsymbol{\omega}}(t)\right){}^I_G\hat{\bar{\mathbf{q}}}(t)\ ,\ {}^G\dot{\hat{\mathbf{p}}}(t) = {}^G\hat{\mathbf{v}}(t)\ ,\ {}^G\dot{\hat{\mathbf{v}}}(t) = {}^G\hat{\mathbf{a}}(t)\\
&\dot{\hat{\mathbf{b}}}_g(t) = \mathbf{0}_{3\times1}\ ,\ \dot{\hat{\mathbf{b}}}_a(t) = \mathbf{0}_{3\times1}\ ,\quad {}^G\dot{\hat{\mathbf{p}}}_f(t) = \mathbf{0}_{3\times1}
\end{aligned}
\tag{5}
$$

where $\hat{\mathbf{a}} = \mathbf{a}_m - \hat{\mathbf{b}}_a$ and $\hat{\boldsymbol{\omega}} = \boldsymbol{\omega}_m - \hat{\mathbf{b}}_g$. The error state of dimension 18×1 is hence defined as follows [see (1)]:

$$
\widetilde{\mathbf{x}}(t) = \left[{}^I\widetilde{\boldsymbol{\theta}}^T(t)\ \widetilde{\mathbf{b}}_g^T(t)\ {}^G\widetilde{\mathbf{v}}^T(t)\ \widetilde{\mathbf{b}}_a^T(t)\ {}^G\widetilde{\mathbf{p}}^T(t)\ {}^G\widetilde{\mathbf{p}}_f^T(t)\right]^T \tag{6}
$$

where we have employed the multiplicative error model for a quaternion [29]. That is, the error between the quaternion $\bar{\mathbf{q}}$ and its estimate $\hat{\bar{\mathbf{q}}}$ is the 3×1 angle-error vector, ${}^I\widetilde{\boldsymbol{\theta}}$, implicitly defined by the error quaternion: $\delta\bar{\mathbf{q}} = \bar{\mathbf{q}} \otimes \hat{\bar{\mathbf{q}}}^{-1} \simeq \begin{bmatrix}\frac{1}{2}{}^I\widetilde{\boldsymbol{\theta}}\\ 1\end{bmatrix}$, where $\delta\bar{\mathbf{q}}$ describes the small rotation that causes the true and estimated attitude to coincide. The advantage of this parametrization permits a minimal representation, 3×3 covariance matrix $\mathbb{E}\left[{}^I\widetilde{\boldsymbol{\theta}}\ {}^I\widetilde{\boldsymbol{\theta}}^T\right]$, for the attitude uncertainty. It is important to note that the orientation error, ${}^I\widetilde{\boldsymbol{\theta}}$, satisfies the following rotation-matrix relation [29]:

$$
\mathbf{C}({}^I_G\bar{\mathbf{q}}) \simeq \left(\mathbf{I}_3 - \lfloor{}^I\widetilde{\boldsymbol{\theta}}\times\rfloor\right)\mathbf{C}({}^I_G\hat{\bar{\mathbf{q}}}) \tag{7}
$$

Now the continuous-time error-state propagation is:

$$
\dot{\widetilde{\mathbf{x}}}(t) = \mathbf{F}_c(t)\widetilde{\mathbf{x}}(t) + \mathbf{G}_c(t)\mathbf{n}(t) \tag{8}
$$

where $\mathbf{n} = \left[\mathbf{n}_g^T\ \mathbf{n}_{wg}^T\ \mathbf{n}_a^T\ \mathbf{n}_{wa}^T\right]^T$ is the system noise, $\mathbf{F}_c$ is the continuous-time error-state transition matrix, and $\mathbf{G}_c$ is the input noise matrix (see [29]):

$$
\mathbf{F}_c = \begin{bmatrix}
-\lfloor\hat{\boldsymbol{\omega}}\times\rfloor & -\mathbf{I}_3 & \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3\\
\mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3\\
-\mathbf{C}^T({}^I_G\hat{\bar{\mathbf{q}}})\lfloor\hat{\mathbf{a}}\times\rfloor & \mathbf{0}_3 & \mathbf{0}_3 & -\mathbf{C}^T({}^I_G\hat{\bar{\mathbf{q}}}) & \mathbf{0}_3 & \mathbf{0}_3\\
\mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3\\
\mathbf{0}_3 & \mathbf{0}_3 & \mathbf{I}_3 & \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3\\
\mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3
\end{bmatrix},\quad
\mathbf{G}_c = \begin{bmatrix}
-\mathbf{I}_3 & \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3\\
\mathbf{0}_3 & \mathbf{I}_3 & \mathbf{0}_3 & \mathbf{0}_3\\
\mathbf{0}_3 & \mathbf{0}_3 & -\mathbf{C}^T({}^I_G\hat{\bar{\mathbf{q}}}) & \mathbf{0}_3\\
\mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{I}_3\\
\mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3\\
\mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3
\end{bmatrix} \tag{9}
$$

The system noise is modelled as zero-mean white Gaussian process with autocorrelation $\mathbb{E}\left[\mathbf{n}(t)\mathbf{n}(\tau)^T\right] = \mathbf{Q}_c\delta(t-\tau)$, which depends on the IMU noise characteristics.

We have presented the continuous-time propagation model using IMU measurements. However, in any practical EKF implementation, the discrete-time state-transition matrix, $\boldsymbol{\Phi}(t_{k+1}, t_k)$, is required in order to propagate the error covariance from time t_k to t_{k+1}. Typically it is found by solving the following matrix differential equation:

$$\dot{\boldsymbol{\Phi}}(t_{k+1}, t_k) = \mathbf{F}_c(t_{k+1})\boldsymbol{\Phi}(t_{k+1}, t_k) \tag{10}$$

with the initial condition $\boldsymbol{\Phi}(t_k, t_k) = \mathbf{I}_{18}$. And its solution has the following structure:

$$\boldsymbol{\Phi}_k := \boldsymbol{\Phi}(t_{k+1}, t_k) = \begin{bmatrix} \boldsymbol{\Phi}_{k,11} & \boldsymbol{\Phi}_{k,12} & \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3 \\ \mathbf{0}_3 & \mathbf{I}_3 & \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3 \\ \boldsymbol{\Phi}_{k,31} & \boldsymbol{\Phi}_{k,32} & \mathbf{I}_3 & \boldsymbol{\Phi}_{k,34} & \mathbf{0}_3 & \mathbf{0}_3 \\ \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{I}_3 & \mathbf{0}_3 & \mathbf{0}_3 \\ \boldsymbol{\Phi}_{k,51} & \boldsymbol{\Phi}_{k,52} & \delta t_k \mathbf{I}_3 & \boldsymbol{\Phi}_{k,54} & \mathbf{I}_3 & \mathbf{0}_3 \\ \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{I}_3 \end{bmatrix} \tag{11}$$

where $\delta t_k = t_{k+1} - t_k$. This matrix (11) can be found either numerically [23, 29] or analytically [6, 17]. Once it is computed, the EKF propagates the error covariance in a standard way [20]:

$$\mathbf{P}_{k+1|k} = \boldsymbol{\Phi}_k \mathbf{P}_{k|k} \boldsymbol{\Phi}_k^T + \mathbf{Q}_{d,k} \tag{12}$$

where $\mathbf{Q}_{d,k}$ is the discrete-time system noise covariance computed as follows:

$$\mathbf{Q}_{d,k} = \int_{t_k}^{t_{k+1}} \boldsymbol{\Phi}(t_{k+1}, \tau)\mathbf{G}_c(\tau)\mathbf{Q}_c\mathbf{G}_c^T(\tau)\boldsymbol{\Phi}^T(t_{k+1}, \tau)d\tau. \tag{13}$$

3.2 *Camera Measurement Model*

The camera observes visual corner features, which are used to concurrently estimate the ego-motion of the sensing platform. Assuming a calibrated perspective camera, the measurement of the feature at time-step k is the perspective projection of the 3D point, ${}^{C_k}\mathbf{p}_f$, expressed in the current camera frame $\{C_k\}$, onto the image plane, i.e.,

$$\mathbf{z}_k = \frac{1}{z_k}\begin{bmatrix} x_k \\ y_k \end{bmatrix} + \mathbf{v}_k \tag{14}$$

$$\begin{bmatrix} x_k \\ y_k \\ z_k \end{bmatrix} = {}^{C_k}\mathbf{p}_f = \mathbf{C}({}^C_I\bar{\mathbf{q}})\mathbf{C}({}^{I_k}_G\bar{\mathbf{q}})\left({}^G\mathbf{p}_f - {}^G\mathbf{p}_k\right) + {}^C\mathbf{p}_I \tag{15}$$

where $\mathbf{v}_k$ is the zero-mean, white Gaussian measurement noise with covariance $\mathbf{R}_k$. In (15), $\{{}^C_I\bar{\mathbf{q}}, {}^C\mathbf{p}_I\}$ is the rotation and translation between the camera and the IMU. This transformation can be obtained, for example, by performing camera-IMU extrinsic calibration offline [21].

For the use of EKF, linearization of (14) yields the following measurement residual [see (6)]:

$$\widetilde{\mathbf{z}}_k = \mathbf{H}_k \widetilde{\mathbf{x}}_{k|k-1} + \mathbf{v}_k = \mathbf{H}_{\mathbf{I}_k} \widetilde{\mathbf{x}}_{I_{k|k-1}} + \mathbf{H}_{\mathbf{f}_k} {}^G\widetilde{\mathbf{p}}_{f_{k|k-1}} + \mathbf{v}_k \tag{16}$$

where the measurement Jacobian $\mathbf{H}_k$ is computed as:

$$\mathbf{H}_k = \left[\mathbf{H}_{\mathbf{I}_k}\ \mathbf{H}_{\mathbf{f}_k}\right] = \mathbf{H}_{\mathbf{proj}} \mathbf{C}({}^C_I\bar{\mathbf{q}}) \left[\mathbf{H}_{\boldsymbol{\theta}_k}\ \mathbf{0}_{3\times 9}\ \mathbf{H}_{\mathbf{p}_k}\ \ \mathbf{C}({}^{I_k}_G\hat{\bar{\mathbf{q}}})\right] \tag{17}$$

$$\mathbf{H}_{\mathbf{proj}} = \frac{1}{\hat{z}_k^2} \begin{bmatrix} \hat{z}_k & 0 & -\hat{x}_k \\ 0 & \hat{z}_k & -\hat{y}_k \end{bmatrix} \tag{18}$$

$$\mathbf{H}_{\boldsymbol{\theta}_k} = \lfloor \mathbf{C}({}^{I_k}_G\hat{\bar{\mathbf{q}}}) \left({}^G\hat{\mathbf{p}}_f - {}^G\hat{\mathbf{p}}_k\right) \times \rfloor \tag{19}$$

$$\mathbf{H}_{\mathbf{p}_k} = -\mathbf{C}({}^{I_k}_G\hat{\bar{\mathbf{q}}}) \tag{20}$$

Once the measurement Jacobian and residual are computed, we can apply the standard EKF update equations to update the state estimates and error covariance [20].

4 Optimal-State-Constraint (OSC)-EKF

It is evident from the preceding section that the visual-inertial SLAM formulation has *quadratic* complexity with respect to the number of features in the state vector (due to the EKF covariance update), which is often too expensive to process as hundreds of thousands of features can be easily detected from images. To address this issue, we adopt the multi-state constraint Kalman filter (MSCKF) framework [22] and perform tightly-coupled VINS over a sliding window of n poses without including features in the state vector, thus having complexity *independent* of the total number of features observed in the environment. The key idea of the proposed OSC-EKF is to form a novel measurement model that utilizes all feature observations available within the sliding window to impose probabilistically optimal constraints between poses, without estimating these features as part of the state vector. In the following we derive in detail this measurement model for the use of EKF.

4.1 State Vector

The state vector at time-step k augments the current IMU state by the past n poses where the images were taken through stochastic cloning [25]:

$$\mathbf{x}_k = \left[\mathbf{x}_{I_k}^T\ \mathbf{y}_{k-1}^T\ \cdots\ \mathbf{y}_{k-n}^T\right]^T \tag{21}$$

where $\mathbf{y}_\ell^T = \left[{}^{I_\ell}_G\bar{\mathbf{q}}^T\ \ {}^G\mathbf{p}_\ell^T\right]$ is the IMU pose (quaternion and position) where the image is recorded at time-step ℓ.

4.2 Propagation

During the propagation, the current state estimates evolve forward in time by integrating (2), while the cloning-state estimates remain static. On the other hand, the augmented covariance is propagated as follows (see (12) and [25]):

$$\mathbf{P}_{k+1|k} = \mathbf{Diag}\left(\boldsymbol{\Phi}_k, \mathbf{I}_{6n}\right) \mathbf{P}_{k|k} \mathbf{Diag}\left(\boldsymbol{\Phi}_k^T, \mathbf{I}_{6n}\right) + \mathbf{Diag}\left(\mathbf{Q}_{d,k}, \mathbf{0}_{6n}\right). \tag{22}$$

4.3 Update

During the update, in order to utilize all the information about the poses, which is encapsulated in all the camera measurements to the features observed in the current sliding window, we aim to derive optimal constraints between camera poses in the window. To this end, we first perform structure and motion [24] using the camera observations *only* (i.e., without using IMU measurements), and then marginalize out the structure, thus resulting in optimal up-to-scale motion constraints.

Consider a feature point, f_j, is observed by the camera at time-step i, for $i = k-n, \ldots, k$, and $j = 1, \ldots, m$ [see (14)].

$$\mathbf{z}_{ij} = \mathbf{h}\left({}^L\mathbf{x}_{C_i}, {}^L\mathbf{p}_{f_j}\right) + \mathbf{n}_{ij} \tag{23}$$

where the measurement noise $\mathbf{n}_{ij} \sim \mathcal{N}(\mathbf{0}, \mathbf{R}_{ij})$, and $\{L\}$ denotes the reconstruction frame which is initialized by two camera views in the current time window $[k-n, k]$ when performing structure and motion and that conforms the corresponding epipolar geometry, and can be simply chosen to be the first frame of the current window.

We formulate the relative maximum likelihood estimation (MLE) using *only* the feature measurements with respect to the *relative* states expressed in the local frame $\{L\}$, i.e., ${}^L\mathbf{x}_\mathbf{C}$ that contains all the camera poses in the time window $[k-n, k]$, and ${}^L\mathbf{p}_\mathbf{F}$ that includes all the feature points. Under the common assumptions of Gaussian noise and independent measurements, this relative MLE is equivalent to the following nonlinear least squares problem:

$$\min_{{}^L\mathbf{x}_\mathbf{C}, {}^L\mathbf{p}_\mathbf{F}} \sum_{i,j} ||\mathbf{z}_{ij} - \mathbf{h}({}^L\mathbf{x}_{C_i}, {}^L\mathbf{p}_{f_j})||^2_{\mathbf{R}_{ij}} \tag{24}$$

where we have employed the energy norm notation, i.e., $||\mathbf{r}||^2_\mathbf{R} = \mathbf{r}^T\mathbf{R}^{-1}\mathbf{r}$. Note that in our implementation the inverse depth parameterization for the feature points is used in order to achieve better numerical stability and thus better estimates [1]. To solve this problem (24), a Gauss-Newton iterative approach is typically employed. The information (Hessian) matrix of the relative camera states (i.e., the relative-pose constraints), is obtained using Schur complement by marginalizing out ${}^L\mathbf{p}_\mathbf{F}$:

$$\mathbf{A} = \sum_{i,j} \mathbf{H}_{ij}^T \mathbf{R}_{ij}^{-1} \mathbf{H}_{ij} =: \begin{bmatrix} \mathbf{A_{FF}} & \mathbf{A_{FC}} \\ \mathbf{A_{CF}} & \mathbf{A_{CC}} \end{bmatrix} \Rightarrow \tag{25}$$

$$\mathbf{A}_p = \mathbf{A_{CC}} - \mathbf{A_{CF}} \mathbf{A_{FF}}^{-1} \mathbf{A_{FC}} \tag{26}$$

where $\mathbf{H}_{ij}$ is the measurement Jacobian with respect to the relative states, ${}^L\mathbf{x_C}$ and ${}^L\mathbf{p_F}$ [see (24)], and the Hessian matrix $\mathbf{A}$ is partitioned based on the dimensions of these two states.

Close inspection reveals that the information matrix $\mathbf{A}$ encapsulates all the information about the relative states, contained in the feature measurements in the current sliding window; and thus $\mathbf{A}_p$, the Schur complement of $\mathbf{A_{FF}}$ in $\mathbf{A}$, describes *all* the information that the feature measurements provide us about the relative camera poses, ${}^L\mathbf{x_C}$. Therefore, the following relative constraints for the camera poses induced from the camera measurements in the current time window, are *optimal* (up to linearization errors):

$${}^L\mathbf{x_C} \sim \mathcal{N}\left({}^L\hat{\mathbf{x}}_C, \mathbf{R}_p\right) \tag{27}$$

where $\mathbf{R}_p = \mathbf{A}_p^{-1}$ is the covariance of inferred relative-pose measurements. Note that, due to marginalization (26), in general, the information matrix $\mathbf{A}_p$, and thus the corresponding covariance matrix $\mathbf{R}_p$, is full, which implies that the induced relative constraints for the remaining states are *correlated*.

Now we can use these inferred optimal relative measurements to update the global state estimates in the EKF [see (21)]. In particular, for each relative camera pose ${}^L\mathbf{x}_{C_i} = \begin{bmatrix} {}^{C_i}_{L}\bar{\mathbf{q}} \\ {}^L\mathbf{p}_{C_i} \end{bmatrix}$, it is not difficult to derive the following nonlinear relations among the global states:

$$\mathbf{C}({}^{C_i}_{L}\bar{\mathbf{q}}) = \mathbf{C}({}^{I}_{C}\bar{\mathbf{q}})^T \mathbf{C}({}^{I_i}_{G}\bar{\mathbf{q}}) \mathbf{C}({}^{I_L}_{G}\bar{\mathbf{q}})^T \mathbf{C}({}^{I}_{C}\bar{\mathbf{q}}) \tag{28}$$

$${}^L\mathbf{p}_{C_i} = \mathbf{C}({}^{I}_{C}\bar{\mathbf{q}})^T \left[\mathbf{C}({}^{I_L}_{G}\bar{\mathbf{q}}) \left(\mathbf{C}({}^{I_i}_{G}\bar{\mathbf{q}})^{T\,I}\mathbf{p}_C + {}^G\mathbf{p}_i - {}^G\mathbf{p}_L\right) - {}^I\mathbf{p}_C\right] \tag{29}$$

For the use of EKF, we need to linearize these equations with respect to the global states (specifically, ${}^{I_L}_{G}\bar{\mathbf{q}}$, ${}^G\mathbf{p}_L$, ${}^{I_i}_{G}\bar{\mathbf{q}}$, and ${}^G\mathbf{p}_i$) around the current state estimates.

First consider the inferred relative rotation (28). Using the small-angle approximation (7), we can rewrite this equation as follows:

$$\begin{aligned}
&\left(\mathbf{I}_3 - \lfloor\tilde{\boldsymbol{\phi}}_{iL}\times\rfloor\right) \mathbf{C}({}^{C_i}_{L}\hat{\bar{\mathbf{q}}}) \\
&= \mathbf{C}({}^{I}_{C}\bar{\mathbf{q}})^T \left(\mathbf{I}_3 - \lfloor\tilde{\boldsymbol{\theta}}_i\times\rfloor\right) \mathbf{C}({}^{I_i}_{G}\hat{\bar{\mathbf{q}}}) \left(\mathbf{I}_3 + \lfloor\tilde{\boldsymbol{\theta}}_L\times\rfloor\right) \mathbf{C}({}^{I_L}_{G}\hat{\bar{\mathbf{q}}})^T \mathbf{C}({}^{I}_{C}\bar{\mathbf{q}}) \\
&= \mathbf{C}({}^{I}_{C}\bar{\mathbf{q}})^T \mathbf{C}({}^{I_i}_{G}\hat{\bar{\mathbf{q}}}) \mathbf{C}({}^{I_L}_{G}\hat{\bar{\mathbf{q}}})^T \mathbf{C}({}^{I}_{C}\bar{\mathbf{q}}) + \mathbf{C}({}^{I}_{C}\bar{\mathbf{q}})^T \mathbf{C}({}^{I_i}_{G}\hat{\bar{\mathbf{q}}}) \mathbf{C}({}^{I_L}_{G}\hat{\bar{\mathbf{q}}})^T \lfloor\tilde{\boldsymbol{\theta}}_L\times\rfloor \mathbf{C}({}^{I}_{C}\bar{\mathbf{q}}) \\
&- \mathbf{C}({}^{I}_{C}\bar{\mathbf{q}})^T \lfloor\tilde{\boldsymbol{\theta}}_i\times\rfloor \mathbf{C}({}^{I_i}_{G}\hat{\bar{\mathbf{q}}}) \mathbf{C}({}^{I_L}_{G}\hat{\bar{\mathbf{q}}})^T \mathbf{C}({}^{I}_{C}\bar{\mathbf{q}}) - \mathbf{C}({}^{I}_{C}\bar{\mathbf{q}})^T \lfloor\tilde{\boldsymbol{\theta}}_i\times\rfloor \mathbf{C}({}^{I_i}_{G}\hat{\bar{\mathbf{q}}}) \mathbf{C}({}^{I_L}_{G}\hat{\bar{\mathbf{q}}})^T \lfloor\tilde{\boldsymbol{\theta}}_L\times\rfloor \mathbf{C}({}^{I}_{C}\bar{\mathbf{q}}) \\
&\simeq \mathbf{C}({}^{C_i}_{L}\hat{\bar{\mathbf{q}}}) + \mathbf{C}({}^{I}_{C}\bar{\mathbf{q}})^T \mathbf{C}({}^{I_i}_{G}\hat{\bar{\mathbf{q}}}) \mathbf{C}({}^{I_L}_{G}\hat{\bar{\mathbf{q}}})^T \lfloor\tilde{\boldsymbol{\theta}}_L\times\rfloor \mathbf{C}({}^{I}_{C}\bar{\mathbf{q}}) - \mathbf{C}({}^{I}_{C}\bar{\mathbf{q}})^T \lfloor\tilde{\boldsymbol{\theta}}_i\times\rfloor \mathbf{C}({}^{I_i}_{G}\hat{\bar{\mathbf{q}}}) \mathbf{C}({}^{I_L}_{G}\hat{\bar{\mathbf{q}}})^T \mathbf{C}({}^{I}_{C}\bar{\mathbf{q}})
\end{aligned} \tag{30}$$

where $\boldsymbol{\phi}_{iL}$ denotes the angle corresponding to the relative rotation $\mathbf{C}({}^{C_i}_{L}\bar{\mathbf{q}})$, and $\widetilde{\boldsymbol{\theta}}_i$ and $\widetilde{\boldsymbol{\theta}}_L$ are the angle errors expressed in the IMU local frame, i.e., $\widetilde{\boldsymbol{\theta}}_i := {}^{I}\widetilde{\boldsymbol{\theta}}_i$ and $\widetilde{\boldsymbol{\theta}}_L := {}^{I}\widetilde{\boldsymbol{\theta}}_L$. We hereafter drop the left superscripts to keep the presentation concise. In the last line we have also neglected the second-order term. From (30), we have:

$$\begin{aligned}
&\mathbf{C}({}^{C_i}_{L}\hat{\bar{\mathbf{q}}})^T \lfloor \widetilde{\boldsymbol{\phi}}_{iL} \times \rfloor \\
&\simeq -\mathbf{C}({}^{I}_{C}\bar{\mathbf{q}})^T \lfloor \widetilde{\boldsymbol{\theta}}_L \times \rfloor \mathbf{C}({}^{I_L}_{G}\hat{\bar{\mathbf{q}}})\mathbf{C}({}^{I_i}_{G}\hat{\bar{\mathbf{q}}})^T\mathbf{C}({}^{I}_{C}\bar{\mathbf{q}}) + \mathbf{C}({}^{I}_{C}\bar{\mathbf{q}})^T\mathbf{C}({}^{I_L}_{G}\hat{\bar{\mathbf{q}}})\mathbf{C}({}^{I_i}_{G}\hat{\bar{\mathbf{q}}})^T \lfloor \widetilde{\boldsymbol{\theta}}_i \times \rfloor \mathbf{C}({}^{I}_{C}\bar{\mathbf{q}}) \\
&= -\lfloor \mathbf{C}({}^{I}_{C}\bar{\mathbf{q}})^T \widetilde{\boldsymbol{\theta}}_L \times \rfloor \mathbf{C}({}^{C_i}_{L}\hat{\bar{\mathbf{q}}})^T + \mathbf{C}({}^{C_i}_{L}\hat{\bar{\mathbf{q}}})^T \lfloor \mathbf{C}({}^{I}_{C}\bar{\mathbf{q}})^T \widetilde{\boldsymbol{\theta}}_i \times \rfloor \\
&= \mathbf{C}({}^{C_i}_{L}\hat{\bar{\mathbf{q}}})^T \lfloor -\mathbf{C}({}^{C_i}_{L}\hat{\bar{\mathbf{q}}})\mathbf{C}({}^{I}_{C}\bar{\mathbf{q}})^T \widetilde{\boldsymbol{\theta}}_L \times \rfloor + \mathbf{C}({}^{C_i}_{L}\hat{\bar{\mathbf{q}}})^T \lfloor \mathbf{C}({}^{I}_{C}\bar{\mathbf{q}})^T \widetilde{\boldsymbol{\theta}}_i \times \rfloor
\end{aligned} \tag{31}$$

Now we have the following error equation, from which we can easily read out the Jocobian matrices with respect to the global orientation states ${}^{I_L}_{G}\bar{\mathbf{q}}$ and ${}^{I_i}_{G}\bar{\mathbf{q}}$.

$$\widetilde{\boldsymbol{\phi}}_{iL} \simeq -\mathbf{C}({}^{I}_{C}\bar{\mathbf{q}})^T\mathbf{C}({}^{I_i}_{G}\hat{\bar{\mathbf{q}}})\mathbf{C}({}^{I_L}_{G}\hat{\bar{\mathbf{q}}})^T\widetilde{\boldsymbol{\theta}}_L + \mathbf{C}({}^{I}_{C}\bar{\mathbf{q}})^T\widetilde{\boldsymbol{\theta}}_i =: \mathbf{H}_{\theta_L}\widetilde{\boldsymbol{\theta}}_L + \mathbf{H}_{\theta_i}\widetilde{\boldsymbol{\theta}}_i \tag{32}$$

where we have used the fact that $\mathbf{C}({}^{C_i}_{L}\hat{\bar{\mathbf{q}}}) = \mathbf{C}({}^{I}_{C}\bar{\mathbf{q}})^T\mathbf{C}({}^{I_i}_{G}\hat{\bar{\mathbf{q}}})\mathbf{C}({}^{I_L}_{G}\hat{\bar{\mathbf{q}}})^T\mathbf{C}({}^{I}_{C}\bar{\mathbf{q}})$.

Now consider the inferred relative position (29). It is important to note that the relative positions between camera poses were determined up to scale (by using only camera measurements). To compensate for this fact, we can normalize the relative positions by the z-coordinate of the k-th relative position, ${}^{L}\mathbf{p}_{C_k}(3)$, i.e.,[2]

$${}^{L}\bar{\mathbf{p}}_{C_i} := \frac{{}^{L}\mathbf{p}_{C_i}}{{}^{L}\mathbf{p}_{C_k}(3)} = \rho\, {}^{L}\mathbf{p}_{C_i} \tag{33}$$

where $\rho := \frac{1}{{}^{L}\mathbf{p}_{C_k}(3)}$. Thus, based on (29), we can linearize this up-to-scale relative-position measurement (33) with respect to the global states around the current state estimates and obtain the following linearized measurement error equation:

$${}^{L}\widetilde{\bar{\mathbf{p}}}_{C_i} \simeq \mathbf{H}_{p_L}\, {}^{G}\widetilde{\mathbf{p}}_L + \mathbf{H}_{p_i}\, {}^{G}\widetilde{\mathbf{p}}_i + \mathbf{H}_{p_k}\, {}^{G}\widetilde{\mathbf{p}}_k + \mathbf{H}_{\theta_L}\widetilde{\boldsymbol{\theta}}_L + \mathbf{H}_{\theta_i}\widetilde{\boldsymbol{\theta}}_i + \mathbf{H}_{\theta_k}\widetilde{\boldsymbol{\theta}}_k \tag{34}$$

where the Jacobian matrices are computed using the chain rule of differentiation, for $i \neq k$, as follows:

$$\mathbf{H}_{p_L} = -\hat{\rho}\mathbf{C}({}^{I}_{C}\bar{\mathbf{q}})^T\mathbf{C}({}^{I_L}_{G}\hat{\bar{\mathbf{q}}}) + {}^{L}\hat{\bar{\mathbf{p}}}_{C_i}\mathbf{J}_{P_L} \tag{35}$$

$$\mathbf{H}_{p_i} = \hat{\rho}\mathbf{C}({}^{I}_{C}\bar{\mathbf{q}})^T\mathbf{C}({}^{I_L}_{G}\hat{\bar{\mathbf{q}}}) \tag{36}$$

$$\mathbf{H}_{p_k} = {}^{L}\hat{\bar{\mathbf{p}}}_{C_i}\mathbf{J}_{P_k} \tag{37}$$

$$\mathbf{H}_{\theta_L} = \hat{\rho}\mathbf{C}({}^{I}_{C}\bar{\mathbf{q}})^T \lfloor \mathbf{C}({}^{I_L}_{G}\hat{\bar{\mathbf{q}}}) \left(\mathbf{C}({}^{i}_{G}\hat{\bar{\mathbf{q}}})^I\mathbf{p}_C + {}^{G}\hat{\mathbf{p}}_i - {}^{G}\hat{\mathbf{p}}_L \right) \times \rfloor + {}^{L}\hat{\bar{\mathbf{p}}}_{C_i}\mathbf{J}_{T_L} \tag{38}$$

[2]While in principle we can choose an arbitrary coordinate of any relative position to normalize in order to compensate for the unknown scale, in practice we may select the one that yields the best numerical stability.

$$\mathbf{H}_{\theta_i} = -\hat{\rho}\mathbf{C}({}^I_C\bar{\mathbf{q}})^T\mathbf{C}({}^{I_L}_G\hat{\bar{\mathbf{q}}})\mathbf{C}({}^{I_i}_G\hat{\bar{\mathbf{q}}})\lfloor{}^I\mathbf{p}_C\times\rfloor \tag{39}$$

$$\mathbf{H}_{\theta_k} = {}^L\hat{\mathbf{p}}_{C_i}\mathbf{J}_{T_k} \tag{40}$$

where ${}^L\hat{\mathbf{p}}_{C_i}$ is computed based on (29) using the current state estimates, and $\hat{\rho} = \frac{1}{{}^L\hat{\mathbf{p}}_{C_k}(3)}$. In the above expressions, the Jacobians of ρ with respect to the pertinent global states are computed using the current state estimates as follows:

$$\mathbf{J}_{P_L} := \frac{\partial\rho}{\partial\mathbf{p}_L}|_{\mathbf{p}_L=\hat{\mathbf{p}}_L} = \hat{\rho}^2\mathbf{e}_3^T\mathbf{C}({}^I_C\bar{\mathbf{q}})^T\mathbf{C}({}^{I_L}_G\hat{\bar{\mathbf{q}}}) \tag{41}$$

$$\mathbf{J}_{P_k} := \frac{\partial\rho}{\partial\mathbf{p}_k}|_{\mathbf{p}_k=\hat{\mathbf{p}}_k} = -\hat{\rho}^2\mathbf{e}_3^T\mathbf{C}({}^I_C\bar{\mathbf{q}})^T\mathbf{C}({}^{I_L}_G\hat{\bar{\mathbf{q}}}) \tag{42}$$

$$\mathbf{J}_{T_L} := \frac{\partial\rho}{\partial\boldsymbol{\theta}_L}|_{\boldsymbol{\theta}_L=\hat{\boldsymbol{\theta}}_L} = -\hat{\rho}^2\mathbf{e}_3^T\mathbf{C}({}^I_C\bar{\mathbf{q}})^T\lfloor\mathbf{C}({}^{I_L}_G\hat{\bar{\mathbf{q}}})\left(\mathbf{C}({}^{I_k}_G\hat{\bar{\mathbf{q}}})^I\mathbf{p}_C + {}^G\hat{\mathbf{p}}_k - {}^G\hat{\mathbf{p}}_L\right)\times\rfloor \tag{43}$$

$$\mathbf{J}_{T_k} := \frac{\partial\rho}{\partial\boldsymbol{\theta}_k}|_{\boldsymbol{\theta}_k=\hat{\boldsymbol{\theta}}_k} = \hat{\rho}^2\mathbf{e}_3^T\mathbf{C}({}^I_C\bar{\mathbf{q}})^T\mathbf{C}({}^{I_L}_G\hat{\bar{\mathbf{q}}})\mathbf{C}({}^{I_k}_G\hat{\bar{\mathbf{q}}})^T\lfloor{}^I\mathbf{p}_C\times\rfloor \tag{44}$$

Note that if $i = k$, the measurement (33) becomes identical to the perspective camera model (14), and thus the corresponding Jacobians can be computed as in (17).

Based on the above novel residual equations (32) and (34) along with the measurement noise covariance $\mathbf{R}_p$ (27), we can perform the standard EKF update [20]. It should be noted again that the same observability-based constraints as in [6, 7, 10] can be enforced in the proposed OSC-EKF when computing the propagation and measurement Jacobians in order to improve estimation consistency.

5 Experimental Results

To prove the concept of the proposed OSC-EKF algorithm, we have preliminarily performed VINS experiments in an indoor envrionment, in which a hand-held IMU-camera platform travelled in various indoor environments. In these experiments, we were using a PointGrey Chameleon monocular camera that records images of resolution 640 × 480 pixels at 30 Hz and a MicroStrain IMU (3DM-GX3-25) which operates at 100 Hz (see Fig. 1). We employed the Shi-Tomasi corner detector [27] to extract point features from the first image available in the current sliding window that consists of $n = 10$ camera poses, and then track them over the subsequent images in the window using the KLT tracking algorithm [18]. On average, approximately $m = 200$ features were tracked per image and as a common practice, outliers were rejected from the resulting tracks by running 2-point RANSAC [30].

In this test, the camera-IMU platform traversed about 22 m and returned to its starting position. Figure 2 shows the estimated trajectory. At the end of the trajectory, the OSC-EKF has a position error of 30 cm, which accounts for approximately 1.3% of the total distance travelled. Note that we found from various experiments that the navigation performance can be greatly affected by sensor calibration and

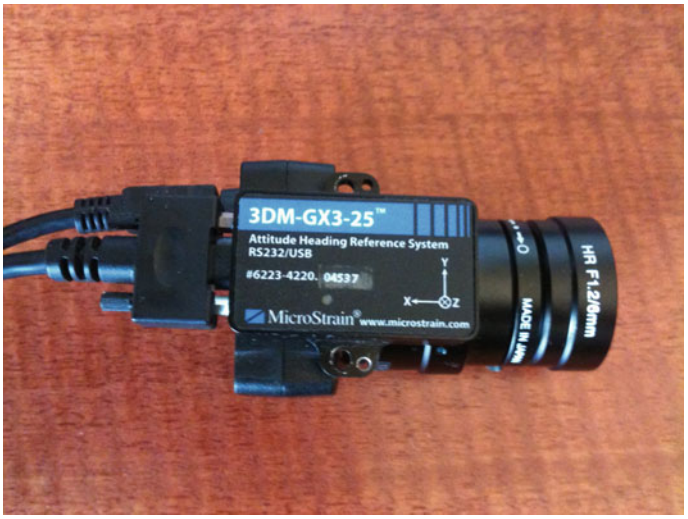
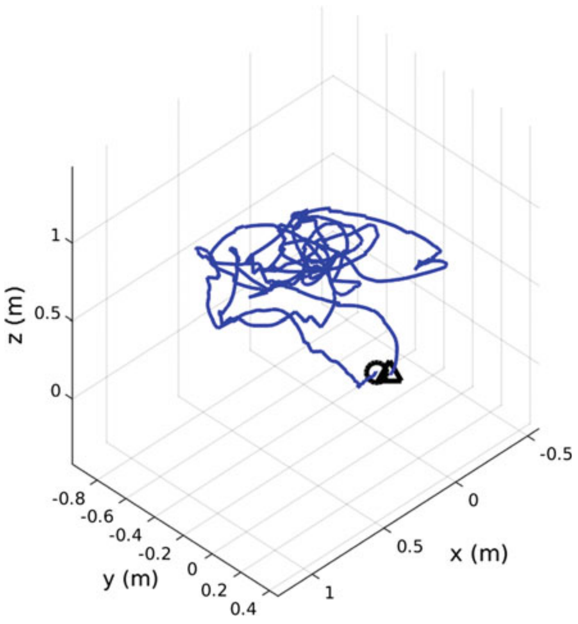

3DM-GX3-25™
Attitude Heading Reference System
RS232/USB
#6223-4220. 04537
MicroStrain® www.microstrain.com
HR F1.2/6mm
MADE IN JAPAN

Fig. 1 The sensor suite used in our experiments: A MicroStrain 3DM-GX3-25 IMU is rigidly mounted on a PointGrey Chameleon monocular camera

Fig. 2 The estimated trajectory of the proposed OSC-EKF in a real-world VINS experiment conducted in a small office room. In this plot, △ denotes the starting position, while the estimated ending position is denoted by ◯



synchronization and KLT-based visual tracking, which we aim to improve in the future. Figure 3 respectively depicts the orientation (roll, pitch and yaw) and velocity (along x, y and z-axis) estimates and their 3σ bounds. These results prove the concept that we seek to promote (i.e., deriving optimal motion constraints for VINS from

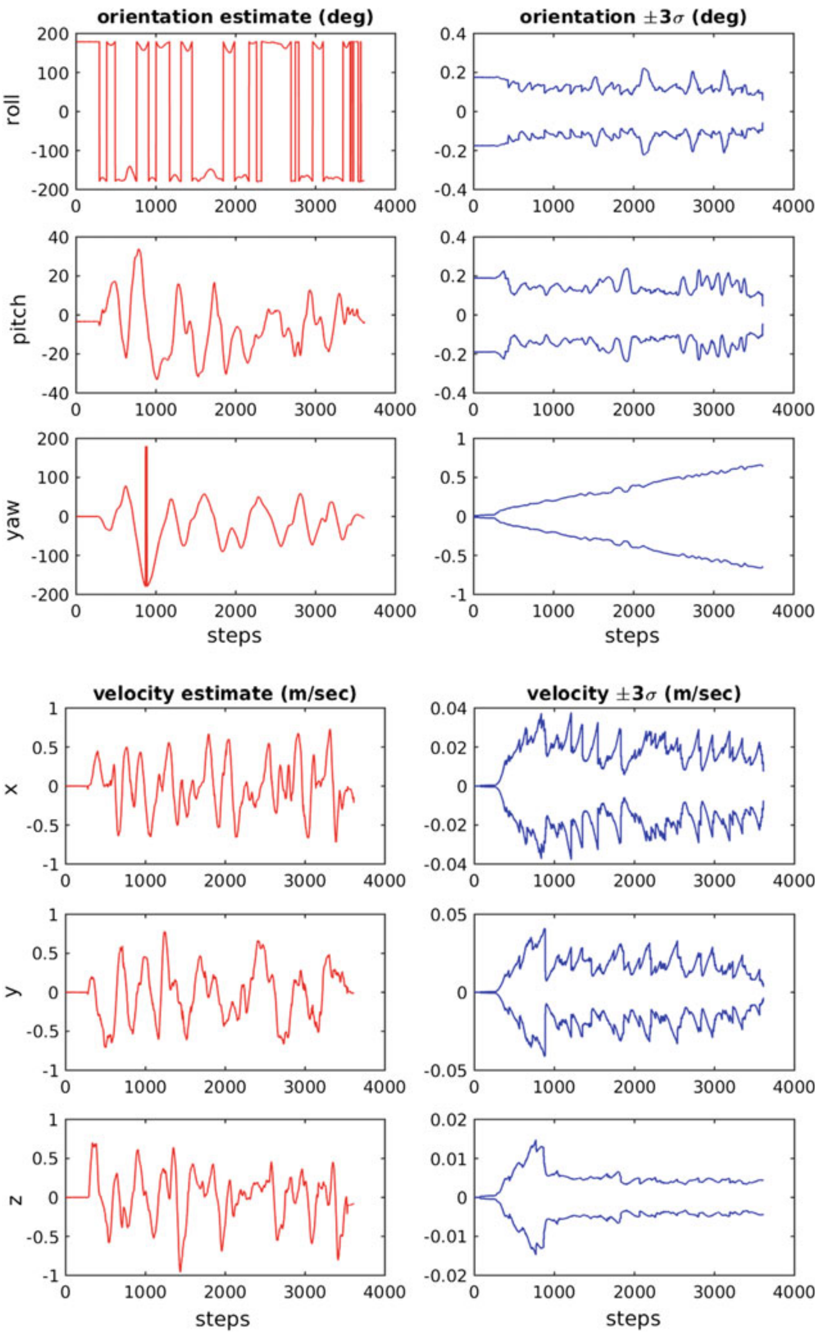


Fig. 3 Orientation and velocity estimates (*left*) and their 3σ bounds (*right*)

feature measurements without including them in the state vector) and show that the proposed approach is able to well track the 6 d.o.f. motion of the sensor suite.

6 Conclusions and Future Work

In this paper, we have introduced an efficient OSC-EKF for visual-inertial navigation, which imposes optimal constraints on the camera poses without estimating visual features as part of the state vector and thus has linear computational complexity in terms of the number of features. In particular, in order to optimally fuse visual and inertial measurements over time, the proposed OSC-EKF maintains a sliding window of poses and derives optimal relative-pose measurements by performing structure and motion with images available in the current window and then marginalizing out the structure. The proposed approach has been preliminarily validated on real-world experiments. In the future, we will test the proposed approach more thoroughly and seek to further improve its performance, e.g., by integrating loop closure to enable long-term navigation while bounding localization errors.

Acknowledgements This work was partially supported by the University of Delaware College of Engineering, the Office of Naval Research (ONR) N00014-10-1-0936, N00014-11-1-0688 and N00014-13-1-0588, and the National Science Foundation (NSF) IIS-1318392 and IIS-1566129.

References

1. Civera, J., Davison, A., Montiel, J.: Inverse depth parametrization for monocular SLAM. IEEE Trans. Robot. **24**(5), 932–945 (2008)
2. Corke, P., Lobo, J., Dias, J.: An introduction to inertial and visual sensing. Int. J. Robot. Res. **26**(6), 519–535 (2007)
3. Ebcin, S., Veth, M.: Tightly-Coupled Image-Aided Inertial Navigation Using the Unscented Kalman Filter. Tech. Rep., Air Force Institute of Technology, Dayton, OH (2007)
4. Google: Google Project Tango. Available: https://www.google.com/atap/projecttango
5. Hernandez, J., Tsotsos, K., Soatto, S.: Observability, identifiability and sensitivity of vision-aided inertial navigation. In: Proceedings of the IEEE International Conference on Robotics and Automation, Seattle, WA, pp. 2319–2325 (2015)
6. Hesch, J., Kottas, D., Bowman, S., Roumeliotis, S.: Consistency analysis and improvement of vision-aided inertial navigation. IEEE Trans. Robot. **30**(1), 158–176 (2013)
7. Hesch, J., Kottas, D., Bowman, S., Roumeliotis, S.: Camera-IMU-based localization: observability analysis and consistency improvement. Int. J. Robot. Res. **33**, 182–201 (2014)
8. Huang, G.: Improving the Consistency of Nonlinear Estimators: Analysis, Algorithms, and Applications. Ph.D. Dissertation, Department of Computer Science and Engineering, University of Minnesota (2012)
9. Huang, G., Mourikis, A.I., Roumeliotis, S.I.: Analysis and improvement of the consistency of extended Kalman filter-based SLAM. In: Proceedings of the IEEE International Conference on Robotics and Automation, Pasadena, CA, pp. 473–479 (2008)
10. Huang, G., Kaess, M., Leonard, J.: Towards consistent visual-inertial navigation. In: Proceedings of the IEEE International Conference on Robotics and Automation, Hong Kong, China, pp. 4926–4933 (2014)

11. Indelman, V., Dellaert, F.: Incremental light bundle adjustment: probabilistic analysis and application to robotic navigation. In: Sun, Y., Behal, A., Chung, C.-K.R. (eds.) New Development in Robot Vision, vol. 23, pp. 111–136. Springer, Berlin (2015)
12. Indelman, V., Roberts, R., Beall, C., Dellaert, F.: Incremental light bundle adjustment. In: Proceedings of the British Machine Vision Conference, Surrey, UK, pp. 1–11 (2012)
13. Indelman, V., Williams, S., Kaess, M., Dellaert, F.: Information fusion in navigation systems via factor graph based incremental smoothing. Robot. Auton. Syst. **61**(8), 721–738 (2013)
14. Jones, E.S., Soatto, S.: Visual-inertial navigation, mapping and localization: a scalable real-time causal approach. Int. J. Robot. Res. **30**(4), 407–430 (2011)
15. Kelly, J., Sukhatme, G.S.: Visual-inertial sensor fusion: localization, mapping and sensor-to-sensor self-calibration. Int. J. Robot. Res. **30**(1), 56–79 (2011)
16. Kim, J., Sukkarieh, S.: Real-time implementation of airborne inertial-SLAM. Robot. Auton. Syst. **55**(1), 62–71 (2007)
17. Li, M., Mourikis, A.: High-precision, consistent EKF-based visual-inertial odometry. Int. J. Robot. Res. **32**(6), 690–711 (2013)
18. Lucas, B., Kanade, T.: An iterative image registration technique with an application to stereo vision. In: Proceedings of the International Joint Conference on Artificial Intelligence, Vancouver, BC, pp. 674–679 (1981)
19. Martinelli, A.: Vision and IMU data fusion: closed-form solutions for attitude, speed, absolute scale, and bias determination. IEEE Trans. Robot. **28**(1), 44–60 (2012)
20. Maybeck, P.S.: Stochastic Models, Estimation, and Control, vol. 1. Academic Press, London (1979)
21. Mirzaei, F.M., Roumeliotis, S.I.: A Kalman filter-based algorithm for IMU-camera calibration: observability analysis and performance evaluation. IEEE Trans. Robot. **24**(5), 1143–1156 (2008)
22. Mourikis, A.I., Roumeliotis, S.I.: A multi-state constraint Kalman filter for vision-aided inertial navigation. In: Proceedings of the IEEE International Conference on Robotics and Automation, Rome, Italy, pp. 3565–3572 (2007)
23. Mourikis, A., Trawny, N., Roumeliotis, S., Johnson, A., Ansar, A., Matthies, L.: Vision-aided inertial navigation for spacecraft entry, descent, and landing. IEEE Trans. Robot. **25**(2), 264–280 (2009)
24. Pollefeys, M.: Visual 3D modeling from images. In: Girod, B., Seidel, H.-P., Magnor, M.A. (eds.) Vision, Modeling, and Visualization. IOS Press, The Netherlands (2004)
25. Roumeliotis, S.I., Burdick, J.W.: Stochastic cloning: a generalized framework for processing relative state measurements. In: Proceedings of the IEEE International Conference on Robotics and Automation, Washington, DC, pp. 1788–1795 (2002)
26. Shen, S., Mulgaonkar, Y., Michael, N., Kumar, V.: Vision-based state estimation for autonomous rotorcraft MAVs in complex environments. In: Proceedings of the IEEE International Conference on Robotics and Automation, Karlsruhe, Germany, pp. 1750–1756 (2013)
27. Shi, J., Tomasi, C.: Good features to track. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Seattle, WA, pp. 593–600 (1994)
28. Strelow, D.: Motion Estimation from Image and Inertial Measurements, Ph.D. dissertation, CMU (2004)
29. Trawny, N., Roumeliotis, S.I.: Indirect Kalman Filter for 3D Attitude Estimation. University of Minnesota, Department of Computer Science & Engineering, Technical Report (2005)
30. Troiani, C., Martinelli, A., Laugier, C., Scaramuzza, D.: 2-point-based outlier rejection for camera-IMU systems with applications to micro aerial vehicles. In: Proceedings of the IEEE International Conference on Robotics and Automation, Hong Kong, China, pp. 5530–5536 (2014)
31. Tsotsos, K., Chiuso, A., Soatto, S.: Robust inference for visual-inertial sensor fusion. In: Proceedings of the IEEE International Conference on Robotics and Automation, Seattle, WA, pp. 5203–5210 (2015)
32. Weiss, S., Siegwart, R.: Real-time metric state estimation for modular vision-inertial systems. In: Proceedings of the IEEE International Conference on Robotics and Automation, Shanghai, China, pp. 4531–4537 (2011)

Part II
Soft Robotics and Natural Machine Motion

Session Summary

Inspired from nature, there has been an increasing interest in the use of soft materials in the design and construction of robots because most of the biological systems rely on deformable structures. Soft robots can be characterized by the mechanical structures with high deformation, stretchability, and elasticity, while being continuum, modular, and often reconfigurable. These aspects are now being replicated in many advanced robotic systems, some of which were presented in this session.

Phillips-Graffllin and Berenson show their recent development in soft body simulation for the purpose of robotic surgery. The simulation environment employs a newly developed soft-rigid interaction model to reproduce the behaviors of soft organs during surgery including the dissection of tissues. The simulation framework is then used for the optimization of control architectures, and the outcome was verified in the real-world platform. Tosun, Jing, Kress-Gazit, and Yim present also a new simulation technique to efficiently develop modular robots. The simulation framework is open-sourced and contains the software library for assistive designs of complex modular robots and their controllers. Sung and Rus show their technological progress of foldable robots. In contrast to conventional foldable robots that usually have problems with the structural strength, the new approach makes use of thicker materials that are laser-cut and glued together with thin films to achieve foldability while maintaining flexibility and modularity. The paper proposed an algorithm to generate cut patterns to automatically design six-legged robots, for example.

Kupcsik and Lee presented a learning algorithm for dynamic object handover. They formulate the problem as contextual policy search, in which the robot learns object handover by interacting with the human. A key challenge in such robot–human interaction is to develop reward function in handover task under noisy human feedback. Preliminary experiments show that the robot learns to hand over a water bottle naturally and that it adapts to the dynamics of human motion.

A group of teams presented a variety of interesting mechanisms inspired by animals' mechanisms and structures. Renda and Seneviratne presented a unified formulation for describing the dynamics aquatic multi-body soft robots. This formulation accounts for the continuum hyperelastic nature of the vehicles and for the interaction with the dense fluid. Soft robotics designs are inspired by the octopus capable of swimming, manipulating, and crawling. A Cosserat-based formalism that combines a Reissner shell model and a finite-strain beam formulation is conceived. Hammond, Wu, and Asada present the design, fabrication, and experimental characterization of soft pneumatic supernumerary robotic (SR) fingers for programmable motion synergies. There are many situations where an extra finger can be helpful in manipulating objects. The pneumatic SR fingers consist of inflatable, rigidizable finger phalanges, and variable stiffness pneumatic bending actuators which are designed and manufactured using soft robot fabrication methods. The SR grasp assist device demonstrates the feasibility of using variable stiffness pneumatic structures to produce grasp synergies. Cheng, Kim, and Desai present a real-time MR image-guided robotic neurosurgery that utilizes a dexterous mesoscale surgical robot that can work in tight spaces. An MR-compatible prototype of the minimally invasive neurosurgical intracranial robot (MINIR-II) is introduced. Each joint of the robot is actuated by an antagonistic pair of shape memory alloy (SMA) spring actuators with integrated water cooling modules. The proposed water-based cooling strategy is designed to improve the cooling rate, and thus the actuation bandwidth of SMA springs.

Ijspeert presents how numerical models and robots can be used to explore the interplay of these four components (CPGs, reflexes, descending modulation, and musculoskeletal system). Inspired by animals' spinal cord circuits that combine reflex loops and central pattern generators (CPGs), ranging from lamprey to human locomotion, a series of models is presented. These models show that the respective roles of these components have changed during evolution with a dominant role of CPGs in lamprey and salamander locomotion, and a more important role for sensory feedback and descending modulation in human locomotion.

A Multi-soft-body Dynamic Model for Underwater Soft Robots

Federico Renda, Francesco Giorgio-Serchi, Frederic Boyer, Cecilia Laschi, Jorge Dias and Lakmal Seneviratne

1 Introduction

Marine operations and the growing needs of the offshore industry require underwater robots to undertake increasingly daunting tasks in always more forbidding environments. Certain scenarios, however, pose such challenges at sea that standard Remotely Operated Vehicles (ROVs) and Autonomous Underwater Vehicles (AUVs) are likely to be unsuitable. An answer to this problem lies in the development of innovative underwater robots which, endowed with augmented manoeuvrability and flexibility, may provide an aid to the ever growing demands of the marine sector.

In recent times underwater robotics has largely benefited from the growing fascination for bioinspired locomotion, because the design of underwater robots can

F. Renda (✉) · J. Dias · L. Seneviratne
KURI, Khalifa University, Abu Dhabi, UAE
e-mail: federico.renda@kustar.ac.ae

J. Dias
e-mail: jorge.dias@kustar.ac.ae

L. Seneviratne
e-mail: lakmal.seneviratne@kustar.ac.ae

F. Giorgio-Serchi · C. Laschi
The BioRobotics Institute, Scuola Superiore Sant'Anna, Pisa, Italy
e-mail: f.serchi@sssup.it

C. Laschi
e-mail: cecilia.laschi@sssup.it

F. Giorgio-Serchi
Fluid Structure Interactions Group, University of Southampton, Southampton, UK

F. Boyer
IRCCyN, Ecole de Mines de Nantes, Nantes, France
e-mail: frederic.boyer@mines-nantes.fr

A. Bicchi and W. Burgard (eds.), *Robotics Research*, Springer Proceedings in Advanced Robotics 2, DOI 10.1007/978-3-319-51532-8_9

profit massively from the investigation of the swimming strategies, hydrodynamics and physiology of such animals. Several examples exist of aquatic organisms which have been taken as the source of inspiration for designing a trustworthy robotic counterpart. The finned and caudal flapping of fish (e.g. [1]) has gathered the most recognition in the scientific community, in part because of the sound understanding of the underlying physics involved in their locomotion [2].

The design criteria for replicating the actuation mechanism which drives the propulsion of fish has, in most cases, entailed the replacement of continuously deforming bodies by reducing the number of degrees-of-freedom (DOF) with a finite sequence of rigid links and joints. However, lately the attempt has been made by [3] to account for the compliant nature of these organisms by resorting to continuous soft structures and actuators. This is one of the few examples where the design principles of soft robotics [4] have been adapted to the aquatic context. In water the hindrances due to the lack of rigid parts are compensated by the support of the dense medium in which the vehicle is immersed, annihilating many of the limits which soft robots are faced with on land. This has encouraged the authors to develop a new breed of aquatic soft robots inspired by the quintessential soft-bodied sea dweller, the octopus.

While the design of Soft Unmanned Underwater Vehicles (SUUVs), may result fairly uncomplicated, their modeling and control is anything but straight forward. Here we present the first example of a cable-actuated, multi-body, aquatic soft robot and introduce a formulation which accounts for the continuum nature of the robotic platform and allows to describe the dynamics of this vehicle while it travels in a quiescent fluid.

1.1 An Aquatic Multi-body Soft Robot

The octopus sports a range of features which are very much sought for in underwater robotics. These include essentially the capability to swim, crawl and manipulate along with an overall remarkable structural compliance. These make the octopus the perfect paradigm of aquatic vehicle. In the scenario of offshore intervention, where complex environments and highly perturbed conditions are the norm, the design criteria borrowed from the soft/bioinspired approach could represent a viable solution to a broad range of tasks which current Unmanned Underwater Vehicles (UUVs) are unfit for.

By taking inspiration from the octopus, the authors have developed a soft-bodied vehicle capable of travelling in water and replicating some of the salient skills of this aquatic organism [5]. The result of this effort is portrayed schematically in Fig. 1. This first prototype of aquatic multi-body soft robot entails a platform composed for as much as 80% in volume of rubber-like materials and capable of manipulating as well as travelling in the aquatic environment either via waterborne pulsed-jet propulsion or legged locomotion.

The robot essentially consist of a central elastic shell, referred to here as the Soft Shell Mantle (SSM), which is designated to performing the pulsed-jet propulsion via

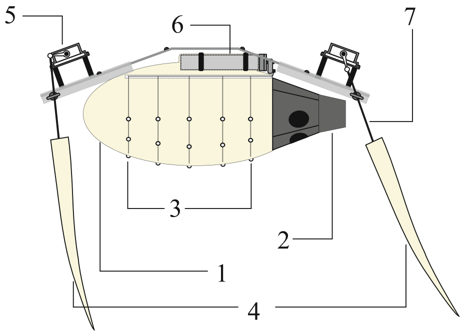


Fig. 1 A schematic of the SUUV developed by the authors. Numbers refer to: (*1*) pulsed-jet thruster, (*2*) the nozzle, (*3*) the cables which drive the shell collapse, (*4*) the continuum manipulators, (*5*) the actuators of the manipulators, (*6*) the actuator of the shell and (*7*) the cable which drives manipulator actuation

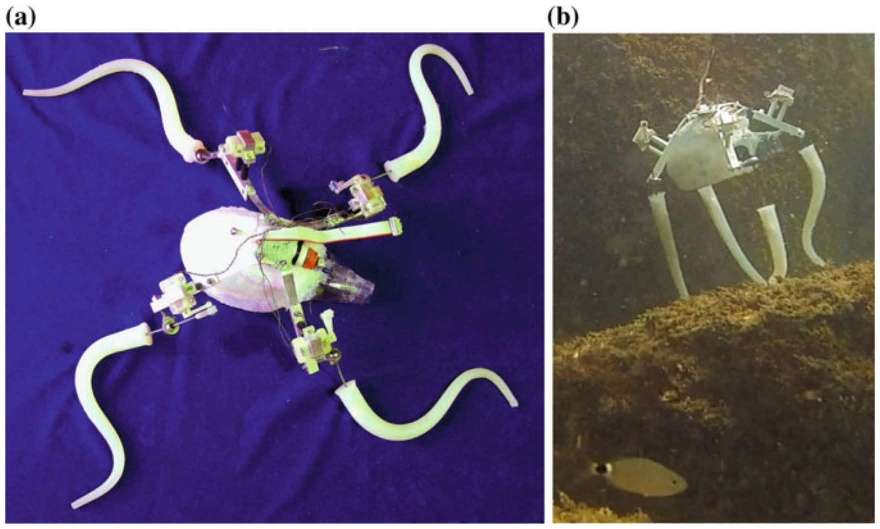


Fig. 2 The Soft Unmanned Underwater Vehicle (SUUV) PoseiDRONE upon assemblage completion (**a**) and during testing at sea (**b**)

the recursive ingestion and expulsion of finite slugs of ambient water [6, 7]. From this central unit, a number of manipulators, i.e. the Soft Robot Arms (SRAs), depart: these are conical-shaped continuous structures composed of elastomeric materials designated to performing basic manipulation [8] and legged-locomotion [9]. Actuation is entirely dealt with via cable-transmission: inextensible cables run through the arms and inside the central shell and, upon recoiling from the designated DC motor, drive the twirling of the manipulator or the collapse of the shell.

Once assembled, the vehicles appears as in Fig. 2a. This vehicle has been tested both in controlled environments as well as in open water, see Fig. 2b and modeling and control of this complex system has been attempted by separately accounting for the various mechanical units [6, 10]. Here, for the first time, the authors attempt to formulate a unified model which encompasses internal actuation and external dynamics.

2 Soft Robot Arm and Soft Shell Mantle Model

In the geometrical exact approach, the SRA is viewed as a Cosserat rod [11], i.e. as a continuous assembly of rigid cross section, while the SSM is modelled as a Cosserat axisymmetric shell [12], i.e. as a continuous assembly of fibers along the median surface. In this section, a brief description of the kinematics and dynamics of a Cosserat rod/shell for underwater soft robotics is given, based on the authors previous works [8, 13], which should be taken as the reference for a more detailed derivation. Experimental validation of the model are also presented in [8] for the SRA, while for the SSM steady state experiments have been presented in [14], dynamic experiments are under review and a coupled dynamic-potential flow solution is given in [15]. In order to appreciate the symmetry between the two models, with a slight abuse of notation, some times we will adopt the same symbols for the two formulations, since they share the same geometrical and mechanical meaning in both the cases.

2.1 *Kinematics*

The reference space is endowed with a base of orthogonal unit vectors $(\mathbf{e}_1, \mathbf{e}_2, \mathbf{e}_3)$ (Fig. 3). In the Cosserat theory, the configuration of a soft body at a certain time is characterized by a position vector $\mathbf{r}$ and a material orientation matrix R, para-

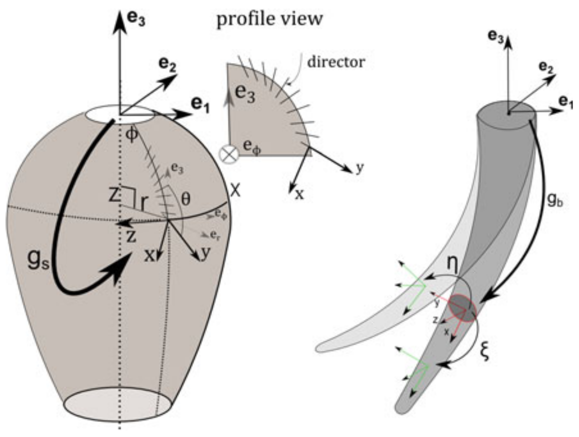


Fig. 3 Sketch of the kinematics which show the geometrical meaning of the elements g, ξ and η. The reference frames on the figure are those used in the model

meterized by the material abscissas, that are $\phi \in [0, 2\pi[$, the angle of revolution of the axisymmetric surface, and $X \in [0, L_s]$ the abscissa along the meridian for the SSM; and $X \in [0, L_b]$, the abscissa along the robot arm, for the SRA (the subscripts $_s$ and $_b$ stand respectively for shell and beam). Thus, the configuration space is defined as a curves $g_b(X)$ and a surface $g_s(X, \phi) \in SE(3)$, with $g_b = \begin{pmatrix} R_b & \mathbf{r}_b \\ 0 & 1 \end{pmatrix}$ and $g_s = \begin{pmatrix} R_s & \mathbf{r_s} \\ 0 & 1 \end{pmatrix}$.

In order to exploit the axisymmetry of the SSM, we introduce another orthogonal basis attached to the material point (X, ϕ): $(\mathbf{e}_r, \mathbf{e}_\phi, \mathbf{e}_3)$ (Fig. 3); defined by roto-translating $(\mathbf{e}_1, \mathbf{e}_2, \mathbf{e}_3)$ of $g_1(X, \phi)$ equal to: $g_1 = \begin{pmatrix} \exp(\tilde{\mathbf{e}}_3\phi) & \mathbf{r}_s \\ 0 & 1 \end{pmatrix}$, where exp is the exponential in SO(3). In this case $\mathbf{r}_s(X)$ take the form: $\mathbf{r}_s = (\cos(\phi)r, \sin(\phi)r, z)^T$ for which, $r(X)$ and $z(X)$ are two smooth functions which define the radius and the altitude of the point X on the profile (Fig. 3). For the sake of convenience, we introduce another orthogonal basis $(\mathbf{e}_r, \mathbf{e}_3, -\mathbf{e}_\phi)$ rotated from the former by $g_2 = \begin{pmatrix} \exp(\tilde{\mathbf{e}}_r\pi/2) & 0 \\ 0 & 1 \end{pmatrix}$. Then, if we call $\theta(X)$ the angle between $\mathbf{e}_3$ and the shell fiber located at any X along the ϕ-meridian, the so called *director orthogonal frame* $(\mathbf{x}, \mathbf{y}, \mathbf{z})$ is defined at each instant t, by $g_3(X)$ equal to: $g_3 = \begin{pmatrix} \exp(-\tilde{\mathbf{e}}_\phi\theta) & 0 \\ 0 & 1 \end{pmatrix}$. Finally, putting them all together, the shell configuration space is $g_s = g_1 g_2 g_3$.

Now, the tangent vector field along the curve $g_b(X)$ is defined by $\widehat{\xi}(X) = g_b^{-1}\partial g_b/\partial X = g_b^{-1}g_b' \in se(3)$ and the tangent plane on the surface $g_s(X, \phi)$ is defined by the two vector fields: $\widehat{\xi_1}(X) = g_s^{-1}\partial g_s/\partial X = g_s^{-1}g_s'$ and $\widehat{\xi_2}(X) = g_s^{-1}\partial g_s/\partial(r^\star\phi) = g_s^{-1}g_s^{\wr}$ (where * denote variable in the reference configuration). In local frame components we have: $\xi = (\mathsf{k}^T, \mathsf{g}^T)^T$, $\xi_1 = (\mathsf{k}_1^T, \mathsf{g}_1^T)^T = (0, 0, \mu, \lambda, \beta, 0)^T$, $\xi_2 = (\mathsf{k}_2^T, \mathsf{g}_2^T)^T = (\frac{\sin(\theta)}{r^\star}, \frac{\cos(\theta)}{r^\star}, 0, 0, 0, -\frac{r}{r^\star})^T \in \mathbb{R}^6$. where $\mathsf{g}(X)$ represents the linear strains, and $\mathsf{k}(X)$ the angular strain. The hat is the isomorphism between the twist vector space $\mathbb{R}^6$ and the Lie algebra $se(3)$.

The time evolution of the configuration curve g_b and surface g_s is represented by the twist vector field $\eta(X) \in \mathbb{R}^6$ defined respectively by $\widehat{\eta_b} = g_b^{-1}\partial g_b/\partial t = g_b^{-1}\dot{g}_b$ and $\widehat{\eta_s} = g_s^{-1}\partial g_s/\partial t = g_s^{-1}\dot{g}_s$. As before, in the local components we have: $\eta_b = (\mathsf{w}_b^T, \mathsf{v}_b^T)^T$, $\eta_s = (\mathsf{w}_s^T, \mathsf{v}_s^T)^T = (0, 0, \Omega, V_x, V_y, 0)^T \in \mathbb{R}^6$, where $\mathsf{v}(X)$ and $\mathsf{w}(X)$ are respectively the linear and angular velocity of a material element at a given instant.

In accordance with this kinematics, the state vector of the SRA and of the SSM are represented by the terns (g_b, ξ, η_b) and (g_s, ξ_1, η_s). From the development above, we can derived the kinematic equations (1) and (2), while in the next sections the compatibility equations and the dynamic equations will be derived (the tilde is the isomorphism between a vector of $\mathbb{R}^3$ and the corresponding skew-symmetric matrix $\in so(3)$).

$$\dot{\mathbf{r}}_b = R_b\mathsf{v}_b \qquad \dot{R}_b = R_b\widetilde{\mathsf{w}}_b. \tag{1}$$

$$\begin{aligned}\dot{\theta} &= \Omega \\ \dot{r} &= \cos(\theta)V_x - \sin(\theta)V_y \\ \dot{z} &= \sin(\theta)V_x + \cos(\theta)V_y\end{aligned} \tag{2}$$

2.2 Strain Measures

There are different ways to measure the strain of a continuous media, we choose the most common used in the specialized literature for the beam and shell separately.

For the SRA, the strains are defined as the difference between the deformed configuration ξ and the reference configuration $\xi^\star$. In particular, the components of $\mathsf{k} - \mathsf{k}^\star$ measure the torsion and the bending state in the two directions. Similarly, the components of $\mathsf{g} - \mathsf{g}^\star$ represent the longitudinal strain (extension, compression) and the two shear strains.

For the SSM, in accordance with [12] as described in [13], the strain tensor field which describes the membrane strain state in the mid-surface is $e(X) = 1/2(h - h^\star)$ where $h(X)$ is the first fundamental form of the Reissner shell equal to $h = diag(\lambda^2 + \beta^2, r^2/r^{\star 2})$. Thus we have $e = (1/2) * diag(\lambda^2 + \beta^2 - 1, r^2/r^{\star 2} - 1)$, in which we have defined $h^\star_{11} = 1$. For what concerns the shear strain state, we have $s(X) = \beta - \beta^\star$. Finally, the flexural strain state is parametrized by the tensor field $d(X) = k - k^\star$, where $k(X)$ is the second fundamental form equal to $k = diag(-\mu\lambda, -r\sin(\theta)/r^{\star 2})$. Thus we have $d = diag(\mu^\star - \mu\lambda, \sin(\theta^\star)/r^\star - r\sin(\theta)/r^{\star 2})$. Furthermore, it is natural to consider that there is no transverse shearing in the reference resting configuration, i.e. $\beta^\star = 0$.

2.3 Compatibility Equations

We have seen above that $g_b' = g_b\widehat{\xi}$ and $g_s' = g_s\widehat{\xi_1}$. By taking the derivative of these equations with respect to time and recalling that $\dot{g} = g\widehat{\eta}$, we obtain the following compatibility equations between velocity and deformation variables: $\dot{\xi} = \eta_b' + \mathrm{ad}_\xi(\eta_b)$ and $\dot{\xi}_1 = \eta_s' + \mathrm{ad}_{\xi_1}(\eta_s)$, where ad is the adjoint map. In local components, we obtain:

$$\begin{aligned}\dot{\mathsf{g}} &= \mathsf{v}_b' + \mathsf{k} \times \mathsf{v}_b - \mathsf{w}_b \times \mathsf{g} \\ \dot{\mathsf{k}} &= \mathsf{w}_b' + \mathsf{k} \times \mathsf{w}_b\end{aligned} \tag{3}$$

$$\begin{aligned}\dot{\mu} &= \Omega' \\ \dot{\lambda} &= V_x' + \beta\Omega - \mu V_y \\ \dot{\beta} &= V_y' - \lambda\Omega + \mu V_x\end{aligned} \tag{4}$$

2.4 Dynamics

The p.d.e.'s describing the evolution of a Reissner rod and shell (not necessarily axisymmetric) have been derived respectively in [11] and [12]. With respect to the local reference frame, these p.d.e's can be written, in a geometric notation, as: $\mathscr{M}_b\dot{\eta}_b = \mathscr{F}'_{bi} + \text{ad}^*_{\xi}(\mathscr{F}_{bi}) + \mathscr{F}_{be} - \text{ad}^*_{\eta_b}(\mathscr{M}_b\eta_b)$ and $\mathscr{M}_s\dot{\eta}_s = 1/j(j\mathscr{F}^1_{si})' + \text{ad}^*_{\xi_\alpha}(\mathscr{F}^\alpha_{si}) + \mathscr{F}_{se} - \text{ad}^*_{\eta_s}(\mathscr{M}_s\eta_s)$, where $j = \sqrt{\det(h)} = r/r^\star\sqrt{\lambda^2 + \beta^2}$, $\mathscr{F}_{bi}(X)$ and $\mathscr{F}^\alpha_{si}(X)$ are the wrenches of internal forces in the surface directions given by g_α (α running over $\{1, 2\}$), $\mathscr{F}_{be}(X)$ and $\mathscr{F}_{se}(X)$ are the external wrench of distributed applied forces, $\mathscr{M}_b(X)$ and $\mathscr{M}_s(X)$ are the screw inertia matrix and $\text{ad}^* = -\text{ad}^T$ is the co-adjoint map. For the repeated α the Einstein convention has to be used as in the rest of the paper. Let us specify the angular and linear components of the internal and external wrenches respectively (for the axisymmetric case see [16]): $\mathscr{F}_{bi} = (\mathsf{M}_b^T, \mathsf{N}_b^T)^T$, $\mathscr{F}^1_{si} = (\mathsf{M}_s^{1T}, \mathsf{N}_s^{1T})^T = (0, 0, M_X, N_X, H, 0)^T$, $\mathscr{F}^2_{si} = (\mathsf{M}_s^{2T}, \mathsf{N}_s^{2T})^T = (M_{\phi_x}, M_{\phi_y}, 0, 0, 0, -N_\phi)^T \in \mathbb{R}^6$, and $\mathscr{F}_{be} = (\mathsf{m}_b^T\mathsf{n}_b^T)^T$, $\mathscr{F}_{se} = (\mathsf{m}_s^T, \mathsf{n}_s^T)^T = (0, 0, l, f_x, f_y, 0)^T \in \mathbb{R}^6$, where $\mathsf{N}(X)$ and $\mathsf{M}(X)$ are the internal force and torque vectors, respectively, while $\mathsf{n}(X)$ and $\mathsf{m}(X)$ are the external force and torque for unit of X. The screw inertia matrices are equal to: $\mathscr{M}_b = \rho_b * diag(I_b, J_b, J_b, A, A, A)$ and $\mathscr{M}_s = \rho_s * diag(J_s, I_s, J_s, 2h_s, 2h_s, 2h_s) \in \mathbb{R}^6 \times \mathbb{R}^6$. In the equations above ρ_b and ρ_s are the body densities, $A(X)$ is the section area equal to $A = \pi h_b^2$, where $h_b(X)$ is the cross section radius, h_s is the half of the shell thickness and $J(X)$, $I(X)$ are the second moment of inertia of the micro-solid equal to $J_b = \pi h_b^4/4$, $J_s = h_s^2/3$, $I_b = \pi h_b^4/2$, $I_s \sim 0$. In components, the dynamic equations are:

$$\begin{aligned} \rho_b A\dot{\mathsf{v}}_b &= \mathsf{N}'_b + \mathsf{k} \times \mathsf{N}_b + \mathsf{n}_b - \mathsf{w}_b \times \rho_b A\mathsf{v}_b \\ \rho_b \mathbf{J}_b\dot{\mathsf{w}}_b &= \mathsf{M}'_b + \mathsf{k} \times \mathsf{M}_b + \mathsf{g} \times \mathsf{N}_b + \mathsf{m}_b - \mathsf{w}_b \times \rho_b\mathbf{J}_b\mathsf{w}_b \end{aligned} \tag{5}$$

$$\begin{aligned} \rho_s J_s\dot{\Omega} &= 1/j(jM_X)' + \lambda H - \beta N_X - \tfrac{\cos(\theta)}{r^\star}M_{\phi_x} + \tfrac{\sin(\theta)}{r^\star}M_{\phi_y} + l \\ 2\rho_s h_s\dot{V}_x &= 1/j(jN_X)' - \mu H - \tfrac{\cos(\theta)}{r^\star}N_\phi + f_x + 2\rho_s h_s\Omega V_y \\ 2\rho_s h_s\dot{V}_y &= 1/j(jH)' + \mu N_X + \tfrac{\sin(\theta)}{r^\star}N_\phi + f_y - 2\rho_s h_s\Omega V_x \end{aligned} \tag{6}$$

where $\mathbf{J}_b$ is equal to $diag(I_b, J_b, J_b)$.

2.5 Constitutive Equations

A linear visco-elastic constitutive equation, based on the Kelvin–Voigt model, is chosen. In [8] and [13] we have found respectively:

$$\mathscr{F}_{bi} = \Sigma(\xi - \xi^\star) + \Upsilon(\dot{\xi}), \tag{7}$$

$$\begin{aligned}
N_X &= \tfrac{2Eh_s}{1-\nu^2}\left[\lambda\left(e_{11}+\nu e_{22}\right)-J_s\mu\left(d_{11}+\nu d_{22}\right)\right] \\
&\quad+\tfrac{6\upsilon h_s}{1-\nu^2}\left[\lambda\left(\dot{e}_{11}+\nu\dot{e}_{22}\right)-J_s\mu\left(\dot{d}_{11}+\nu\dot{d}_{22}\right)\right] \\
N_\phi &= \tfrac{2Eh_s}{1-\nu^2}\left[\tfrac{r}{r^\star}\left(e_{22}+\nu e_{11}\right)-J_s\tfrac{\sin(\theta)}{r^\star}\left(d_{22}+\nu d_{11}\right)\right] \\
&\quad+\tfrac{6\upsilon h_s}{1-\nu^2}\left[\tfrac{r}{r^\star}\left(\dot{e}_{22}+\nu\dot{e}_{11}\right)-J_s\tfrac{\sin(\theta)}{r^\star}\left(\dot{d}_{22}+\nu\dot{d}_{11}\right)\right] \\
H &= 2h_s\beta\left[G+\tfrac{E}{1-\nu^2}\left(e_{11}+\nu e_{22}\right)\right]+2h_s\dot{\beta}\left[\upsilon+\tfrac{3\upsilon}{1-\nu^2}\left(\dot{e}_{11}+\nu\dot{e}_{22}\right)\right] \\[1ex]
M_X &= -\tfrac{2Eh_sJ_s}{1-\nu^2}\lambda\left(d_{11}+\nu d_{22}\right)-\tfrac{6\upsilon h_sJ_s}{1-\nu^2}\lambda\left(\dot{d}_{11}+\nu\dot{d}_{22}\right) \\
M_{\phi_x} &= -\tfrac{2Eh_sJ_s}{1-\nu^2}\tfrac{r}{r^\star}\left(d_{22}+\nu d_{11}\right)-\tfrac{6\upsilon h_sJ_s}{1-\nu^2}\tfrac{r}{r^\star}\left(\dot{d}_{22}+\nu\dot{d}_{11}\right) \\
M_{\phi_y} &= 0
\end{aligned} \tag{8}$$

where $\Sigma(X)$ and $\Upsilon(X) \in \mathbb{R}^6 \otimes \mathbb{R}^6$ are the screw stiffness matrix and the screw viscosity matrix, equal to $\Sigma = diag(GI_b, EJ_b, EJ_B, EA, GA, GA), \Upsilon = \upsilon * diag(I_b, 3J_b, 3J_b, 3A, A, A)$, where E is the Young modulus, G is the shear modulus (equal to $G = E/2(1+\nu)$ for an isotropic material with Poisson ratio ν) and υ is the shear viscosity modulus.

2.6 External Loads

The external loads taken into account are the ones exerted by the fluid (i.e. drag, added mass, buoyancy and thrust) in addition to the gravity load. Mathematically we have:

$$\mathsf{n}_b = \mathsf{gr}_b + \mathsf{b}_b + \mathsf{d}_b + \mathsf{a}_b \tag{9}$$

$$\mathsf{n}_s = \mathsf{d}_s + \mathsf{a}_s + \mathsf{t}_s \tag{10}$$

where $\mathsf{gr}_b(X)$ is the gravity, $\mathsf{b}_b(X)$ is the buoyancy, $\mathsf{t}_s(X)$ is the thrust load, $\mathsf{d}(X)$ is the drag and $\mathsf{a}(X)$ is the added mass.

An exhaustive derivation and interpretation of the fluid force model for the SSM is today under review, based on the usual model of net external forces exerted on a rigid rocket, uniformly "rubbed on" the mantle. Here only the final equation are reported, since it does not affect the scope of the present work. For the SRA, the fluid force models have been originally derived in [17] and then introduced in a soft robotics content in [8].

Gravity and buoyancy are simply the product between the mass per unit of X of the robot arm ρ_b and of the water ρ_w respectively, and the gravity acceleration $gr = -9.81$: $\mathsf{gr}_b + \mathsf{b}_b = (\rho_b - \rho_w)AR_b^T\mathbf{G}$, where $\mathbf{G}$ is the gravity acceleration vector, equal to $\mathbf{G} = (0, 0, gr)^T$.

The drag load vector is proportional to the square of the velocity vector and is directed in the opposite direction. The amplitude of the drag load is also determined by the geometry of section X and by hydrodynamics phenomena expressed by empirical coefficients. For the SRA and the SSM respectively we have: $\mathsf{d}_b = -\rho_w \mathsf{v}_b^T \mathsf{v}_b D \dfrac{\mathsf{v}_b}{|\mathsf{v}_b|}$

and $\mathsf{d}_s = R_s^T(0, 0, -\frac{\rho_w C_d A_{ref} V|V|}{2A_m})^T$, where $D(X) \in \mathbb{R}^3 \otimes \mathbb{R}^3$ is equal to $D = h_b * diag(\frac{1}{2}\pi C_x, C_y, C_z)$ for circular cross sections of radius h_b, C_x, C_y, C_z being the empirical hydrodynamic coefficients; A_{ref} is the reference area equal to $\pi(max(r(X)))^2$, A_m is the total surface of the SSM and C_d is the net drag coefficient. V is the swimming velocity calculated at every time step as the average of the scalar field $\dot{z}(X)$, i.e. $V = (1/A_m)\int_o^{L_s}\int_0^{2\pi}\dot{z}(-z^{\star'})dXr^{\star}d\phi$.

The added mass load vector is proportional to the acceleration vector and is directed in the opposite direction. The amplitude is also determined by the geometry of section X and by hydrodynamics phenomena expressed in part by correction coefficients. For the SRA and the SSM respectively we have: $\mathsf{a}_b = -\frac{d(\rho_w F\mathsf{v}_b)}{dt} = -\rho_w F\dot{\mathsf{v}}_b - \mathsf{w}_b \times \rho_w F\mathsf{v}_b$ and $\mathsf{a}_s = -B_s\rho_s 2h_s\dot{\mathsf{v}}_s = -B_s\rho_s 2h_s(\dot{V}_x, \dot{V}_y, 0)^T$, where B_s is the net added mass coefficient and $F(X) \in \mathbb{R}^3 \otimes \mathbb{R}^3$ is a tensor which incorporates the geometric and hydrodynamics factors, equal to $F = diag(0, AB_b, AB_b)$, B_b being the hydrodynamic correction coefficients.

The thrust load is: $\mathsf{t}_s = R_s^T(0, 0, -\frac{\rho_w \dot{U}|\dot{U}|}{A_n A_m})^T$ where A_n is the nozzle area equal to $A_n = \pi h_n^2$ for the outflow and equal to $A_n = 3\pi h_n^2$ for the inflow (where three inlets and one outlet have been used with radius h_n) and U is the mantle inner volume.

2.7 Internal Actuation

In order to actuate the SRA and the SSM, we impose an internal distributed wrench ($\mathscr{F}_a(X,t)$) which represents the input of the model. It can be thought to be the action of the muscle fiber of the body for living organism or the result of embedded cable-driven actuation as in [8]. The final dynamics equations are as follows: $\mathscr{M}_b\dot{\eta} = \mathscr{F}_{bi}' + \mathrm{ad}_\xi^*(\mathscr{F}_{bi}) + \mathscr{F}_{ba} + \mathscr{F}_{be} - \mathrm{ad}_\eta^*(\mathscr{M}_b\eta)$ and $\mathscr{M}_S\dot{\eta} = 1/j(j\mathscr{F}_{si}^1)' + \mathrm{ad}_{\xi_\alpha}^*(\mathscr{F}_{si}^\alpha) + \mathscr{F}_{sa} + \mathscr{F}_{se} - \mathrm{ad}_\eta^*(\mathscr{M}_s\eta)$.

3 Multi-soft-body Dynamic Model

In order to model and control the behavior of a SUUV like the one in Fig. 1, a method to connect together SRAs and SSMs is needed. In this section, the link between the soft bodies in a star configuration (i.e. a tree structure with a mobile base) is shown, then the updating of the external loads of the soft bodies, due to the net motion, is discussed and finally the overall dynamics of the system is modeled.

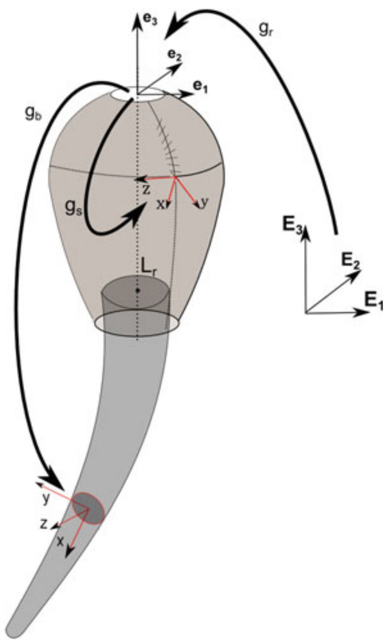


Fig. 4 illustrative scheme of the SUUV kinematics, where $(\mathbf{E}_1, \mathbf{E}_2, \mathbf{E}_3)$ is the Euclidean fixed frame, $(\mathbf{e}_1, \mathbf{e}_2, \mathbf{e}_3)$ represent the rigid body and is the soft bodies reference frame and $(\mathbf{x}, \mathbf{y}, \mathbf{z})$ is the local frame for the μsolids

3.1 Star System Kinematics

In order to link the soft bodies in a star configuration, one can pair a Gathering Rigid Body (GRB) with the already introduced frame $(\mathbf{e}_1, \mathbf{e}_2, \mathbf{e}_3)$. Let us call $(\mathbf{E}_1, \mathbf{E}_2, \mathbf{E}_3)$ the Euclidean fixed frame, hence the GRB configuration space is defined as a point $g_r \in SE(3)$, mapping $(\mathbf{E}_1, \mathbf{E}_2, \mathbf{E}_3)$ in $(\mathbf{e}_1, \mathbf{e}_2, \mathbf{e}_3)$ (Fig. 4), with $g_r = \begin{pmatrix} R_r & \mathbf{r}_r \\ 0 & 1 \end{pmatrix}$.

The time evolution of g_r is represented by the twist vector $\eta_r \in \mathbb{R}^6$ defined by $\widehat{\eta_r} = g_r^{-1}\dot{g_r}$. In accordance to this kinematics, the state vector of the GRB is given by the pair (g_r, η_r) which gives immediately the kinematics equation:

$$\dot{g_r} = g_r \widehat{\eta_r} \tag{11}$$

At this point, the configuration of every point of the SUUV is given by $g = g_r g_{(s,b)}$ as illustrated in Fig. 4. It is worth to notice here that the number, geometry and relative position of the soft bodies can be chosen arbitrary in this scheme. As a matter of fact, in this example, we impose a rigid translation between a SSM and a SRA of L_r (i.e. the length of the GRB), by adding $-L_r$ to the third element of $\mathbf{r}_b$ (Fig. 4).

3.2 External Loads Update

In order to take into account the overall motion, at each time step, the state vector of the GRB (g_r, η_r) is exploited to calculate the actual value of the external loads of the soft bodies (10), (9). In our case, we have rotated the gravity vector $\mathbf{G}$ of R_r^T obtaining the following new equation: $\mathsf{gr}_b + \mathsf{b}_b = (\rho_b - \rho_w) A R_b^T R_r^T \mathbf{G}$ Then, we have updated the swimming velocity V and the linear velocity v_b, by adding respectively the scalar $V_r = (0, 0, 0, 0, 0, 1)\eta_r$ (i.e. the linear velocity of the GRB in the swimming direction $\mathbf{e}_3$) and the vector $\bar{\mathsf{v}}_r = diag(0, 0, 0, 1, 1, 1)\mathrm{Ad}_{g_b^{-1}}\eta_r$ (i.e. the linear velocity of the GRB transported in the local reference frame $(\mathbf{x}, \mathbf{y}, \mathbf{z})$), where Ad is the Adjoint map. The new drag load equations for the SRA and for the SSM become: $\mathsf{d}_b = -\rho_w(\mathsf{v}_b + \bar{\mathsf{v}}_r)^T(\mathsf{v}_b + \bar{\mathsf{v}}_r) D \dfrac{\mathsf{v}_b + \bar{\mathsf{v}}_r}{|\mathsf{v}_b + \bar{\mathsf{v}}_r|}$ and $\mathsf{d}_s = R_s^T(0, 0, -\frac{\rho_w C_d A_{ref}(V+V_r)|V+V_r|}{2A_m})^T$.

3.3 Star System Dynamics

At this point, to obtain the dynamics of the SUUV, we only miss the one of the Gathering Rigid Body that collect the soft appendices. In a geometric notation, it can be written as:

$$\mathscr{M}_r \dot{\eta}_r = \mathscr{F}_r - \mathrm{ad}^*_{\eta_r}(\mathscr{M}_r \eta_r) \tag{12}$$

The soft bodies of the SUUV collected together by the GRB, are frozen in their current shape. By taking advantage of that, the above unknown parameters $(\mathscr{M}_r, \mathscr{F}_r)$ can be calculated as follow ([18]): $\mathscr{M}_r = \int_0^{2\pi} \int_0^{L_s} \mathrm{Ad}^*_{g_s} \mathscr{M}_s \mathrm{Ad}_{g_s^{-1}}(-z^{\star'}) dX r^\star d\phi + \int_0^{L_b} \mathrm{Ad}^*_{g_b} \mathscr{M}_b \mathrm{Ad}_{g_b^{-1}} dX + \mathscr{M}_{ri} = \mathscr{M}'_s + \mathscr{M}'_b + \mathscr{M}_{ri}$ and $\mathscr{F}_r = \int_0^{2\pi} \int_0^{L_s} \mathrm{Ad}^*_{g_s} \mathscr{F}_{se}(-z') dX r d\phi + \int_0^{L_b} \mathrm{Ad}^*_{g_b} \mathscr{F}_{be} \sqrt{\mathsf{g}^T \mathsf{g}} dX + \mathscr{F}_{re} = \mathscr{F}'_{se} + \mathscr{F}'_{be} + \mathscr{F}_{re}$, where $\mathscr{M}_{ri}$ and $\mathscr{F}_{re}$ are respectively the intrinsic inertia and external load directly belonging to the GRB, and $\mathrm{Ad}^*_g = (\mathrm{Ad}_g)^{-T}$ is the coAdjoint map. It is worth to notice that the internal reaction and actuation of the soft bodies does not take part of these integrals, since a frozen shape have to be considered. Furthermore, as a first approximation, the inertia loads due to the relative acceleration of the soft bodies has not been taken into account. The contribute of the added mass loads of the soft bodies in $\mathscr{F}'_{se}$ and $\mathscr{F}'_{be}$ will appear as an additional mass as follow: $\mathscr{M}'_{sa} = B_s \rho_s 2h \int_0^{2\pi} \int_0^{L_s} \mathrm{Ad}^*_{g_s} diag(0, 0, 0, 1, 1, 1) \mathrm{Ad}_{g_s^{-1}}(-z^\star) dX r^\star d\phi$ and $\mathscr{M}'_{ba} = B_b \rho_w A \int_0^{L_b} \mathrm{Ad}^*_{g_b} diag(0, 0, 0, 0, 1, 1) \mathrm{Ad}_{g_b^{-1}} dX$.

Going forward into details, in our case, the intrinsic inertia of the GRB is equal to $\mathscr{M}_{ri} = \rho_r U_r \begin{pmatrix} diag(J_r, J_r, I_r) & \tilde{\mathbf{u}} \\ \tilde{\mathbf{u}}^T & diag(1, 1, 1) \end{pmatrix}$, where $\mathbf{u} = (0, 0, -3L_r/4)^T$ is the position vector of the center of mass of the GRB whit respect to the reference frame $(\mathbf{e}_1, \mathbf{e}_2, \mathbf{e}_3)$; J_r, I_r are the second moment of inertia equal to $J_r = 3(h_r^2/4 + L_r^2)/5$, $I_r = 3h_r^2/10$ (a conic shape have been chosen with base radius h_r), ρ_r is the density and U_r is the volume of the rigid body equal to $U_r = \pi h_r^2 L_r/3$. On the other side, the external loads on the GRB that have been considered are the gravity and buoyancy of the rigid body as well as the gravity and buoyancy of the SSM, since the letter have not been taken into account for the axisymmetric model. Thus we have: $\mathscr{F}_{re} = [(1-\rho_w/\rho_r)\mathscr{M}_{ri} + (1-\rho_w/\rho_s)\mathscr{M}_s']\mathrm{Ad}_{g_r^{-1}}(0,0,0,\mathbf{G}^T)^T$.

3.4 SUUV Dynamic Model

The final system of equations is composed by the second order partial differential equations of the soft bodies and the ordinary differential equations of the star system. The system of p.d.e.'s is composed by the kinematics equation (2), (1), the compatibility equations (4), (3) and the dynamic equations (6), (5) respectively complemented with the internal stresses (8), (7) and the external loads (10), (9). The system of o.d.e.'s for the star system is composed by the kinematic equation (11) and the dynamic equation (12). Finally, in the state form $\dot{x} = f(x, x', x'', t)$, the SUUV model is:

$$\begin{aligned}
&\dot{\theta} = \Omega \\
&\dot{r} = \cos(\theta)V_x - \sin(\theta)V_y \\
&\dot{z} = \sin(\theta)V_x + \cos(\theta)V_y \\
&\dot{\mathbf{r}}_b = R_b \mathsf{v}_b \\
&\dot{R}_b = R_b \widetilde{\mathsf{w}}_b \\
&\dot{g}_r = g_r \widehat{\eta}_r \\
&\dot{\mu} = \Omega' \\
&\dot{\lambda} = V_x' + \beta\Omega - \mu V_y \\
&\dot{\beta} = V_y' - \lambda\Omega + \mu V_x \\
&\dot{\mathsf{k}} = \mathsf{w}_b' + \mathsf{k} \times \mathsf{w}_b \\
&\dot{\mathsf{g}} = \mathsf{v}_b' + \mathsf{k} \times \mathsf{v}_b - \mathsf{w}_b \times \mathsf{g} \\
&\dot{\Omega} = [(jM_X)'/j + \lambda H - \beta N_X - \cos(\theta)M_\phi/r^\star]/(\rho J_s) \\
&\dot{V}_x = [(jN_X)'/j - \mu H - \cos(\theta)N_\phi/r^\star + 2\rho_s h_s \Omega V_y + f_x]/[2h_s\rho_s(1+B_s)] \\
&\dot{V}_y = [(jH)'/j + \mu N_X + \sin(\theta)N_\phi/r^\star - 2\rho_s h_s \Omega V_x + f_y]/[2h_s\rho_s(1+B_s)] \\
&\dot{\mathsf{v}}_b = (\mathsf{N}_b' + \mathsf{k} \times \mathsf{N}_b + \mathsf{n}_b - \mathsf{w}_b \times \rho_b A \mathsf{v}_b)/(\rho_b A) \\
&\dot{\mathsf{w}}_b = (\mathbf{J}_b^{-1}/\rho_b)(\mathsf{M}_b' + \mathsf{k} \times \mathsf{M}_b + \mathsf{g} \times \mathsf{N}_b + \mathsf{m}_b - \mathsf{w}_b \times \rho_b J_b \mathsf{w}_b) \\
&\dot{\eta}_r = \mathscr{M}_r^{-1}[\mathscr{F}_r - \mathrm{ad}^*_{\eta_r}(\mathscr{M}_r \eta_r)]
\end{aligned} \tag{13}$$

The final system is infinite dimensional since all its components are some functions of the profile abscissa X. As a result, in order to be solved numerically, it has to be first space-discretised on a grid of nodes before being time integrated using explicit or implicit time integrators starting from the initial state. In this grid, all the space derivatives appearing in the p.d.e.'s system can be approximated by finite difference schemes, with the following boundary conditions: $\eta(0) = 0$ and $\mathscr{F}_{bi}(L_b) = \mathscr{F}_{si}^1(L_s) = 0$. These operations have been implemented in Matlab©.

4 Results

Although the final goal of this work is to model and control SUUVs like the one in Fig. 1, whereby experimental comparison are needed (as has been done separately for the SRA [8] and for the SSM [14]), in this section an illustrative example, based on simulation, of the setting developed above is presented, in order to demonstrate the feasibility of the proposed mathematical framework to achieve the objective. Further simulation analysis and experimental comparisons with the real prototype, are planned for the extension of the present paper. Finally, an energetic analysis disclosed by the current formulation is conducted and used to describe the results.

4.1 Simulation

One SRA and one SSM have been used, the former has a conical shape with a radius linearly decreasing from $max(h_b(X)) = 15$ to $min(h_b(X)) = 6$ mm, the latter is a semi-sphere of radius 31 mm glued with a cylinder of length 86 mm, both with an half thickness of $h_s = 1$ mm. The GRB has a conical shape too, with a base radius equal to $h_r = max(h_b(X)) = 15$ mm and an height of $L_r = 112$ mm. The density of these bodies has been chosen equal to the one of the water (1022 [kg/m^3]), which makes the structure neutral underwater. The geometrical and mechanical parameters are summarized in Table 1.

In order to reproduce the jet propulsion of the mantle, the SSM is actuated through a triangular wave force function $f_s(X, t)$, perpendicular to the axis of symmetry (thus along $\mathbf{e}_r$), with period T and amplitude ranging in the interval $[F_{min}, F_{max}]$. This pressure has been applied to a central strip of the mantle of height 80 mm to reproduce the bending/steering capability of the robot arm, the SRA is actuated through a linear torque function $f_b(X, t)$ with extremes $[M_{min}, M_{max}]$, directed toward the local axis $\mathbf{z}$ for a certain interval Δt_1 and toward the direction $\mathbf{y}$ for another interval Δt_3, preceded and followed by a rest period of respectively Δt_2 and Δt_4. In other words, the internal distributed wrench $\mathscr{F}_{sa}(X, t)$ takes the form: $\mathscr{F}_{sa} = \text{Ad}^*_{g_3^{-1}}(0, 0, 0, f_s, 0, 0)^T$, and the internal distributed wrench $\mathscr{F}_{ba}(X, t)$ takes different forms for each interval Δt_1, Δt_2, Δt_3, Δt_4, respectively: $\mathscr{F}_{ba} = (0, 0, f_b, 0, 0, 0)^T$, $\mathscr{F}_{ba} = (0, 0, 0, 0, 0, 0)^T$, $\mathscr{F}_{ba} = (0, f_b, 0, 0, 0, 0)^T$, $\mathscr{F}_{ba} = (0, 0, 0, 0, 0, 0)^T$. The loading and dynamic

Table 1 Geometrical and mechanical parameters of the SUUV

	E (kPa)	υ (Pa $*$ s)	ν (−)	ρ (kg/m^3)	L (mm)	h (mm)	h_n (mm)
SSM	40	500	0	1022	130	1	10
SRA	110	300	0	1022	420	[6, 15]	–
GRB	–	–	–	1022	112	15	–

parameters are summarized in Table 2, while a few snapshots of the resulting swimming dynamics is depicted in Fig. 5.

Although a simple actuation pattern has been applied, a complex swimming dynamics came out from the fluid-structure interaction, with unexpected turning around the symmetry axis of the shell mantle ($\mathbf{e}_3$) due to a torsional torque occurred during the switching from one bending to the other. This represents a further motivation toward the development of a proper model of the SUUVs dynamics, in order to be able to understand, design and control these promising devices for underwater exploration.

4.2 Energetic Analysis

Any kind of locomotion is the result of the dynamic interaction between the body deformation and the environment [19]. The quality of this interaction can be measured by calculating how the internal actuation power is translated into the kinetic power of the star system. In the present work, the internal actuation power have been approximate with the positive part of the stress power, reflecting that the elastic energy is mainly due to this action and that it can be only increased by the internal actuation here. With this quantitative index, one can play with the geometrical, mechanical and actuation parameters during the design phase in order to find the best solution for a given application. This is probably one of the killer application of the model.

To do so, we resort to the following efficiency index computed at each time step throughout the simulation $E(t) : E = \frac{W_o}{W_i}$, where $W_o(t)$ and $W_i(t)$ respectively represent the output kinetic power of the star system and the input actuation power. Calculating them we obtain: $W_o = \eta_r^T \mathsf{M}_r \dot{\eta}_r$ and $W_i = W_{is} + W_{ib} = \int\limits_0^{L_s} \int\limits_0^{2\pi}$ $\{\frac{2Eh_s}{1-\nu^2}[E_{11}\dot{E}_{11} + E_{22}\dot{E}_{22} + J_s(D_{11}\dot{D}_{11} + D_{22}\dot{D}_{22})] + 2h_s G\beta\dot{\beta}\}(-z')dXrd\phi + \int\limits_0^{L_b}$ $(\xi - \xi^\star)^T \Sigma \dot{\xi} \sqrt{\mathsf{g}^T \mathsf{g}} dX$, where $W_{is}(t)$ and $W_{ib}(t)$ are respectively the actuation contribution of the SSM and of the SRA (equal to the positive part of the internal elastic energy and to zero in the other case) and we have defined $E_{11} = e_{11} + \nu e_{22}$, $E_{22} = e_{22} + \nu e_{11}$, $D_{11} = d_{11} + \nu d_{22}$ and $D_{22} = d_{22} + \nu d_{11}$. In Fig. 6, the actuation and kinetic power corresponding to the simulation of Fig. 5 are shown. The mean value of the index E is around 3%.

In Fig. 6 can be recognized the different phases of the motion of Fig. 5. In the first part (Δt_1), the input power is composed by the cyclic contraction of the SSM and by the bending of the SRA. In the second phase (Δt_2), the output kinetic power benefits from the released elastic energy of the SRA and from the positive asset of the new configuration, giving a larger gap with respect to the actuation power. In the third part (Δt_3), the actuation power increase due to the new bending and in the last phase

Table 2 Loading and dynamic parameters of the SUUV

	$[F_{min}, F_{max}]$ (Pa)	T (s)	$[M_{min}, M_{max}]$ (N)	$(\Delta t_1, \Delta t_2, \Delta t_3, \Delta t_4)$ (s)	B (−)	C_d (−)	$C_{(x,y,z)}$ (−)
SSM	[0, 5]	0.66	–	–	1.1	1.7	–
SRA	–	–	[0, 0.05]	(1.5, 3, 1.5, 3)	1.5	–	(0.01, 1.5, 1.5)

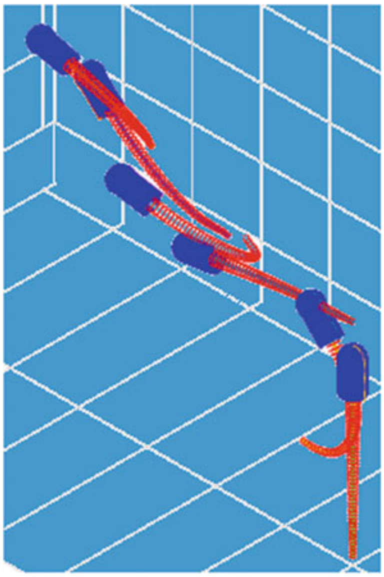


Fig. 5 Few snapshots of the simulation results at $t = 0$, $t = 1.5$, $t = 4$, $t = 5.5$, $t = 7.5$ and $t = 9\,\mathrm{s}$ of the swimming dynamics. The complete video is presented in the media file attached to this paper

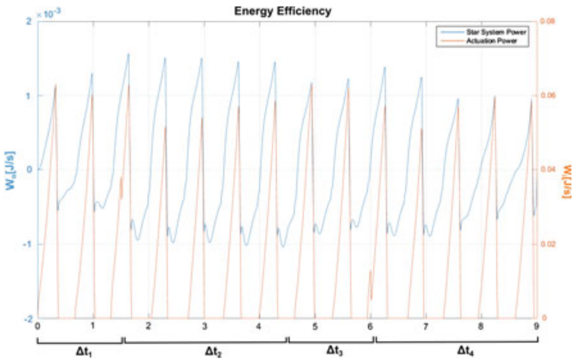


Fig. 6 Kinematic (*blue*) and actuation (*red*) power (note the different scales). The four phases of the simulation of Fig. 5 are highlighted

(Δt_4) the SUUV start to turn around it self due to the hydrodynamic coupling of the bent SRA which is not as efficient as the forward pushing of the second phase, giving a smaller gap between the kinetic and actuation power.

5 Conclusion

A unified model which account for the continuum nature of a multi-soft-body robot has been presented which allows to describe the dynamics of an underwater vehicle while it travels in a quiescent fluid. The potentialities of the model have been demonstrated through an illustrative example, which shows complex and unexpected dynamics of the robot despite of a regular actuation input, underlining how much the behavior of this new kind of systems can be rich and challenging to control. Furthermore, an energetic analysis which takes into account the internal actuation and the mechanical properties of the vehicle is proposed. The work presented here represents a first step toward the development of a mathematical tool for the design and control of intelligent SUUVs.

Acknowledgements This work was partly supported by the Foundation Grant project PoseiDRONE of the Fondazione Livorno.

References

1. Saimek, S., Li, P.Y.: Motion planning and control of a swimming machine. Int. J. Robot. Res. **23**, 27–53 (2004)
2. Colgate, J.E., Lynch, K.M.: Mechanics and control of swimming: a review. IEEE J. Ocean. Eng. **29**, 660–673 (2004)
3. Marchese, A.D., Onal, C.D., Rus, D.: Autonomous soft robotic fish capable of escape maneuvers using fluidic elastomer actuators. Soft Robot. **1**, 75–87 (2014)
4. Trimmer, B.A.: Soft robots. Curr. Biol. **23**, 639–641 (2013)
5. Arienti, A., Calisti, M., Giorgio-Serchi, F., Laschi, C.: PoseiDRONE: design of a soft-bodied ROV with crawling, swimming and manipulation ability. In: Proceedings of the MTS/IEEE OCEANS conference, San Diego, CA, USA, September, 21–27 (2013)
6. Giorgio-Serchi, F., Arienti, A., Baldoli, I., Laschi, C.: An elastic pulsed-jet thruster for Soft Unmanned Underwater Vehicles. In: 2013 IEEE International Conference on Robotics and Automation (ICRA), pp. 5103, 5110, 6–10 May 2013
7. Giorgio-Serchi, F., Arienti, A., Laschi, C.: A soft unmanned underwater vehicle with augmented thrust capability. In: Proceedings of the MTS/IEEE OCEANS conference, San Diego, CA, USA, September, 21–27 (2013)
8. Renda, F., Giorelli, M., Calisti, M., Cianchetti, M., Laschi, C.: Dynamic model of a multibending soft robot arm driven by cables. IEEE Trans. Robot. **30**(5), 1109–1122 (2014)
9. Calisti, M., Giorelli, M., Levy, G., Mazzolai, B., Hochner, B., Laschi, C., Dario, P.: An octopus-bioinspired solution to movement and manipulation for soft robots. Bioinspiration Biomim. **6**, 1–10 (2011)
10. Giorelli, M., Giorgio-Serchi, F., Arienti, A., Laschi, C.: Forward speed control of a pulsed-jet soft-bodied underwater vehicle. In: Proceedings of the MTS/IEEE OCEANS conference, San Diego, CA, USA, September, 21–27 (2013)
11. Simo, J.C.: A finite strain beam formulation. The three dimensional dynamic problem: part I. Comput. Methods Appl. Mech. Eng. **49**, 55–70 (1985)
12. Simo, J.C., Fox, D.D.: On stress resultant geometrically exact shell model. Part I: formulation and optimal parametrization. J. Comput. Methods Appl. Mech. Eng. **72**(3), 267–304 (1989)
13. Renda, F., Giorgio-Serchi, F., Boyer, F., Laschi, C.: Structural dynamics of a pulsed-jet propulsion system for underwater soft robots. Int. J. Adv. Robot. Syst. **12**(68) (2015)
14. Renda, F., Giorgio-Serchi, F., Boyer, F., Laschi, C.: Locomotion and elastodynamics model of an underwater shell-like soft robot. In: 2015 IEEE International Conference on Robotics and Automation (ICRA), pp. 1158, 1165, 26–30 May 2015, Seattle, USA, May 26–30 (2015)
15. Giorgio-Serchi, F., Renda, F., Calisti, M., Laschi, C.: Thrust depletion at high pulsation frequencies in underactuated, soft-bodied, pulsed-jet vehicles. In: MTS/IEEE OCEANS, Genoa, Italy, May 19–21 (2015)
16. Nonlinear Problems of Elasticity. Applied Mathematical Sciences, vol. 107, 2nd edn. Springer, New York (2005)

17. Boyer, F., Porez, M., Khalil, W.: Macro-continuous computed torque algorithm for a three-dimensional eel-like robot. IEEE Trans. Robot. **22**(4), 763–775 (2006)
18. Boyer, F., Ali, S., Porez, M.: Macrocontinuous dynamics for hyperredundant robots: application to kinematic locomotion bioinspired by elongated body animals. IEEE Trans. Robot. **28**(2), 303–317 (2012)
19. Boyer, F., Belkhiri, A.: Reduced locomotion dynamics with passive internal DoF: application to nonholonomic and soft robotics. IEEE Trans. Robot. **30**(3), 578–592 (2014)

Learning Dynamic Robot-to-Human Object Handover from Human Feedback

Andras Kupcsik, David Hsu and Wee Sun Lee

1 Introduction

In the near future, robots will become trustworthy helpers of humans, performing a variety of services at homes and in workplaces. A basic, but essential capability for such robots is to fetch common objects of daily life, e.g., cups or TV remote controllers, and hand them to humans. Today robots perform object handover in a limited manner: typically the robot holds an object statically in place and waits for the human to take it. This is far from the fluid handover between humans and is generally inadequate for the elderly, the very young, or the physically weak who require robot services. The long-term goal of our research is to develop the algorithmic framework and the experimental system that enable robots to perform *fluid* object handover in a *dynamic* setting and to adapt over human preferences and object characteristics. This work takes the first step and focuses on a robot handing over a water bottle in a dynamic setting (Fig. 1), e.g., handing over flyers to people walking by or handing over water bottles to marathon runners.

Object handover appears deceptively simple. Humans are experts at object handover. We perform it many times a day almost flawlessly without thinking and *adapt* over widely different contexts:

- *Dynamics*: We hand over objects to others whether they sit, stand, or walk by.
- *Object characteristics*: We hand over objects of different shape, weight, and surface texture.

A. Kupcsik (✉) · D. Hsu · W.S. Lee
School of Computing, National University of Singapore, Singapore, Singapore
e-mail: kupcsik@comp.nus.edu.sg

D. Hsu
e-mail: dyhsu@comp.nus.edu.sg

W.S. Lee
e-mail: leews@comp.nus.edu.sg

A. Bicchi and W. Burgard (eds.), *Robotics Research*, Springer Proceedings in Advanced Robotics 2, DOI 10.1007/978-3-319-51532-8_10

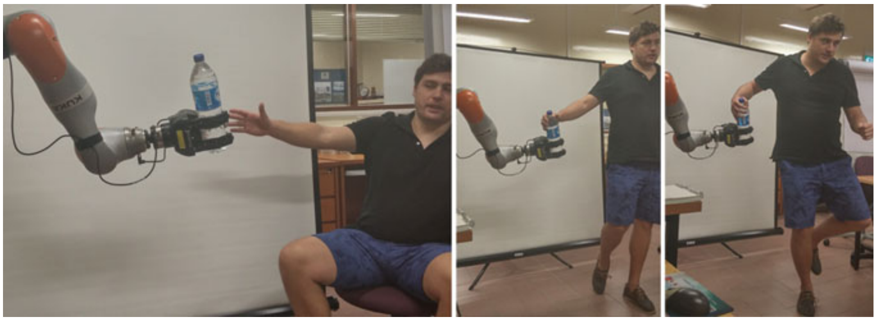


Fig. 1 Hand over a water bottle to a person sitting, walking, or running

- *Human preferences*: While typical human object handover occurs very fast, we adapt our strategy and slow down when handing over objects to the elderly or young children.

The success of humans, however, belies the complexity of object handover as collaborative physical interaction between two agents with limited communication. Manually programming robot handover with comparable robustness and adaptivity poses great challenge, as we lack even a moderately comprehensive and reliable model for handover in a variety of contexts.

Alternatively, the robot can learn the handover skill by interacting with the human and generalize from experience. In this work, we formulate the learning task as *contextual policy search* [19]. Policy search is a general approach to reinforcement learning and has been very successful in skill learning for robot with many degrees of freedom [11]. Policy search algorithms parametrize robot control policies and search for the best parameter values by maximizing a reward function that captures the policy performance. Contextual policy search introduces a set of *context variables* that depend on the task context, e.g., object type or size for the handover task, and the policy parameters are conditioned on the context variables.

A reward function that accurately measures policy performance is key to the success of policy search. However, handcrafting a good reward function is often tedious and error-prone, in particular, for learning object handover. It is unclear what quantitative measures capture fluid object handover. Instead, we propose to learn the latent reward function from human feedback. Humans are experts at object handover and can easily provide reward feedback. However, the feedback is often noisy. To be robust against noise and avoid overfitting, we apply a Bayesian optimization approach to latent reward learning. Importantly, our learning algorithm allows for both *absolute feedback*, e.g., "Is the handover good or bad?", and *preference feedback*, e.g., "Is the handover better than the previous one?". Combining latent reward learning and policy search leads to a holistic contextual policy search algorithm that learns object handover directly from human feedback. Our preliminary experiments show that the robot learns to hand over a water bottle naturally and that it adapts to the dynamics of human motion.

2 Related Work

2.1 *Object Handover*

Object handover has intrigued the research community for a long time from the both physical and social-cognitive perspectives. Early work on handover dates back to at least 1990s [1, 21]. Recent work suggests that object handover consists of three stages conceptually: approach, signal, and transfer [26]. They do not necessarily occur sequentially and may partially overlap. In the first stage, the giver approaches the taker and poses the object to get ready for handover [4, 20, 25]. In the second stage, the giver and taker signal to each other and exchange information, often through non-verbal communication, such as motion [12], eye gaze, or head orientation [13], in order to establish joint intention of handover. In the final stage, they complete the physical transfer of the object. The transfer stage can be further divided into two sub-stages, before and after the giver and the taker establish joint contact of the object, respectively. Earlier work on object transfer generally assumes that the object remains stationary once joint contact is established and relies on handcrafted controllers [1, 6, 14, 21]. Our work focuses to the final physical transfer stage only. The algorithm learns a controller directly from human feedback. It does not make the stationary assumption and caters for dynamic handover. Object transfer is an instance of the more general problem of cooperative manipulation [3]: it involves two asymmetric agents with limited communication.

Human-human object handover provides the yardstick for handover performance. Understanding how humans perform handover (e.g., [5, 15]) paves the way towards improved robot handover performance.

2.2 *Policy Search*

Robot skill learning by policy search has been highly successful in recent years [11]. Policy search algorithms learn a skill represented as a probability distribution over parameterized robot controllers, by maximizing the expected reward. To allow robot skills to adapt to different situations, contextual policy search learns a contextual policy that conditions a skill on context variables [8, 9, 19].

To represent robot skills, policy search typically makes use of parametrized controllers, such as *dynamic movement primitives* [16] or *interaction primitives* [2]. The latter is well-suited for human-robot interaction tasks. Our work, on the other hand, exploits domain knowledge to construct a parameterized impedance controller.

To learn robot skills, policy search requires that a reward function be given to measure learning performance. However, handcrafting a good reward function is often difficult. One approach is inverse reinforcement learning (IRL), also called inverse optimal control, which learns a reward function from expert demonstration [22, 24]. Demonstrations by human experts can be difficult or tedious to acquire, in particular, for robot-human object handover. An alternative is to learn directly from human feedback, without human expert demonstration. Daniel et al. use reward feedback from humans to learn manipulation skills for robot hands [10]. Wilson et al. consider learning control policies from trajectory preferences using a Bayesian approach without explicit reward feedback [27]. Jain et al. learn manipulation trajectories from human preferences [17]. Preference-based reinforcement learning algorithms generally do not use absolute reward feedback and rely solely on preference feedback [28]. Our algorithm combines both absolute and preference feedback in a single Bayesian framework to learn a reward function and integrate with policy search for robot skill learning.

3 Learning Dynamic Handover from Human Feedback

3.1 Overview

Assume that a robot and a human have established the joint intention of handover. Our work addresses the physical transfer of an object from the robot to the human. The robot controller $u(\cdot\,; \boldsymbol{\omega})$ specifies the control action u_t at the state $\boldsymbol{x}_t$ at time t for $t = 1, 2, \ldots$. The controller $u(\cdot\,; \boldsymbol{\omega})$ is parametrized by a set of parameters $\boldsymbol{\omega}$, and the notation makes the dependency on $\boldsymbol{\omega}$ explicit. A reward function $R(\boldsymbol{\omega})$ assigns a real number that measures the performance of the controller $u(\cdot\,; \boldsymbol{\omega})$. To handle the dynamics of handover, we introduce a context variable $\boldsymbol{s}$ representing the velocity of the human hand and condition the controller parameters $\boldsymbol{\omega}$ on $\boldsymbol{s}$, giving rise the reward function $R(\boldsymbol{\omega}, \boldsymbol{s})$. In general, context variables may include other features, such as human preferences and object characteristics as well. A contextual policy $\pi(\boldsymbol{\omega}|\boldsymbol{s})$ is a probability distribution over parametrized controllers, conditioned on the context $\boldsymbol{s}$. Our goal is to learn a contextual policy that maximizes the expected reward:

$$\pi^* = \arg\max_{\pi} \int_{\boldsymbol{s}} \int_{\boldsymbol{\omega}} R(\boldsymbol{\omega}, \boldsymbol{s}) \pi(\boldsymbol{\omega}|\boldsymbol{s}) \mu(\boldsymbol{s}) \, \mathrm{d}\boldsymbol{\omega} \, \mathrm{d}\boldsymbol{s}, \tag{1}$$

where $\mu(\boldsymbol{s})$ is a given prior distribution over the contexts.

Contextual policy search iteratively updates π so that the distribution peaks up on controllers with higher rewards. In each iteration, the robot learner observes context $\boldsymbol{s}$ and samples a controller with parameter value $\boldsymbol{\omega}$ from the distribution $\pi(\cdot|\boldsymbol{s})$. It executes the controller $u(\cdot|\boldsymbol{\omega})$ and observes the reward $R(\boldsymbol{\omega}, \boldsymbol{s})$. After repeating this experiment L times, it updates π with the gathered data $\{\boldsymbol{\omega}_i, \boldsymbol{s}_i, R(\boldsymbol{\omega}_i, \boldsymbol{s}_i)\}_{i=1}^{L}$ and proceeds to the next iteration. See Fig. 2 for the overall learning and control architecture and Table 1 for a sketch of our learning algorithm.

The reward function $R(\boldsymbol{\omega}, \boldsymbol{s})$ is critical in our algorithm. Unfortunately, it is difficult to specify manually a good reward function for learning object handover, despite the many empirical studies of human-human object handover [4, 5, 15, 26]. We

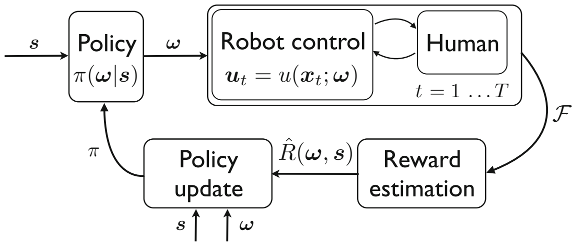


Fig. 2 The human-robot handover skill learning framework. The robot observes context $\boldsymbol{s}$, then samples $\boldsymbol{\omega}$ using the policy $\pi(\boldsymbol{\omega}|\boldsymbol{s})$. The experiment is executed with a robot controller with parametrization $\boldsymbol{\omega}$. The robot controller $u(\boldsymbol{x}; \boldsymbol{\omega})$ provides deterministic control signals $\boldsymbol{u}$ given the state of the robot and its environment $\boldsymbol{x}$. After the experiment the human provides a high-level feedback $\mathscr{F}$, which is used the estimate the latent reward $\hat{R}(\boldsymbol{\omega}, \boldsymbol{s})$. Finally, the policy is updated with the latest data

Table 1 The learning framework for human-robot object transfer

```
The C-REPS Algorithm with Human Feedback
Input: relative entropy bound $\epsilon$, initial policy $\pi(\omega|s)$, maximum number of policy updates $H$.
for $h = 1, \dots, H$
    Collect human feedback data:
        Observe context $s_i \sim \mu(s)$, $i = 1, \dots, L$
        Draw parameters $\omega_i \sim \pi(\omega|s_i)$
        Collect human feedback $\mathscr{F}_i$
    Estimate latent rewards of all previously seen samples $\{\omega_i, s_i, \mathscr{F}_i\}_{i=1}^{E}$
    Predict rewards:
        Draw context $s_j \sim \hat{\mu}(s)$, $j = 1, \dots, Q$
        Draw parameters $\omega_j \sim \pi(\omega|s_j)$
        Predict $\hat{R}(\omega_j, s_j)$ with reward model
    Update policy:
        Update policy $\pi(\omega|s)$ using C-REPS with samples $\{\omega_j, s_j, \hat{R}(\omega_j, s_j)\}_{j=1}^{Q}$
    end
Output: policy $\pi(\omega|s)$
```

propose to learn a reward function $\hat{R}(\boldsymbol{\omega}, \boldsymbol{s})$ from human feedback. Specifically, we allow both *absolute* and *preference* human feedback. Absolute feedback provides direct assessment of the robot controller performance on an absolute scale from 1 to 10. Preference feedback compares one controller with another relatively. While the former has higher information content, the latter is usually easier for humans to assess. We take a Bayesian approach and apply Gaussian process regression to latent reward estimation. The learned reward model $\hat{R}(\boldsymbol{\omega}, \boldsymbol{s})$ generalizes the human feedback data. It provides estimated reward on arbitrarily sampled $(\boldsymbol{\omega}, \boldsymbol{s})$ without additional experiments and drastically reduces the number of robot experiments required for learning a good policy.

3.2 Representing the Object Handover Skill

In this section we discuss how we encode the handover skill and which parameters $\boldsymbol{\omega}$ refers to. In our work we use a trajectory generator, a robot arm controller and a robot hand controller to encode the handover skill. A trajectory generator provides reference Cartesian coordinates for the robot end-effector to follow. In robot learning tasks, Movement Primitives (MP) are often used to encode a trajectory with a limited amount of parameters. MPs encode the shape, speed and magnitude of the trajectory in Cartesian space, or in joint space for each degree of freedom. While MPs can encode a wide variety of skills, they typically require a higher number of parameters to tune, which might slow down the learning process.

For handover tasks however, we can use human expert knowledge to define robot hand trajectories. This approach allows for a more compact representation of the trajectory generator with less parameters to tune. Furthermore, we can address safety by reducing the workspace of the robot and we can easily synchronize with the human motion. In our experiments we use visual data of a Kinect sensor, which tracks the right hand of the human. As soon as the human hand is within d_{max} distance from the robot hand the robot moves the object towards the human hand location. We assume that a path planner computes the reference trajectory from the current robot hand location to the human hand location. The reference trajectory is updated every time the human hand location is updated. As soon as the distance between the human and the robot hand falls below d_{min}, we do not use visual information due to possible occlusion and measurement error. Instead, we use the recorded visual data to predict the human hand trajectory for the next second when the physical interaction is likely to happen. The values of d_{min} and d_{max} may depend on different factors, such as, experiment setup, robot configuration, etc.

In order to ensure robust human-robot handover, we need to allow compliant robot arm motion. We use Cartesian impedance control [3] where the wrench $\boldsymbol{F}_{6\times1}$ concatenating forces and torques exerted in the end-effector frame is computed according to $\boldsymbol{F} = \boldsymbol{M}\Delta\ddot{\boldsymbol{x}} + \boldsymbol{D}\Delta\dot{\boldsymbol{x}} + \boldsymbol{P}\Delta\boldsymbol{x}$, where $\Delta\boldsymbol{x}_{6\times1}$ is the deviation from the reference trajectory. The gain parameters $\boldsymbol{M}$, $\boldsymbol{D}$ and $\boldsymbol{P}$ will determine the amount of exerted forces and torques. $\boldsymbol{M}$ is typically replaced with the robot inertia at the current state. We choose the damping $\boldsymbol{D}$ such that the closed loop control system is critically damped. We use a diagonal stiffness matrix $\boldsymbol{P} = \text{diag}([\boldsymbol{p}_t^T, \boldsymbol{p}_r^T])$, where $\boldsymbol{p}_t$ is the translational and $\boldsymbol{p}_r$ is the rotational stiffness. Finally, the applied torque commands are $\boldsymbol{\tau} = \boldsymbol{J}^T\boldsymbol{F} + \boldsymbol{\tau}_{ff}$, where $\boldsymbol{J}$ is the Jacobian of the robot and $\boldsymbol{\tau}_{ff}$ are feed forward torques compensating for gravity and other nonlinear effects.

Motivated by recent work in human-human handover experiments [5], a robot grip force controller [6] has been proposed $\boldsymbol{F}_g = k\boldsymbol{F}_l + \boldsymbol{F}_{ovl}$, where $\boldsymbol{F}_g$ is the commanded grip force, $\boldsymbol{F}_l$ is the measured load force and $\boldsymbol{F}_{ovl}$ is the preset overloading force. The slope parameter k depends on object properties, such as mass, shape and material properties. When using this controller, the robot will release the object in case the total load force on the robot drops below a threshold value. For robot hands with only finger position control we cannot use the above control approach. Instead, we directly command finger positions by identifying the finger position with minimal grip force that still holds the object. Then, we use a control law to change finger positions linear in the load force $\boldsymbol{f}_{pos} = \boldsymbol{f}_{min} + m\boldsymbol{F}_l$. The value of m depends on many factors, such as, object type, weight and other material properties.

For learning the object handover, we tune 7 parameters of the control architecture. For trajectory generator we tune the minimal and maximal tracking distances d_{min} and d_{max}. For the compliant arm controller we learn the translational stiffness parameters and one parameter for all the rotational stiffness values. Finally, for finger controller we tune the slope parameter. All these parameters are collected in $\boldsymbol{\omega}_{7\times1}$.

3.3 Estimating the Latent Reward Function

In this section we propose a Bayesian latent reward estimation technique based on previous work [7]. Assume that we have observed a set of samples $\{\boldsymbol{s}_i, \boldsymbol{\omega}_i\}_{i=1}^E$ and human feedback $\{\mathscr{F}_i\}_{i=1}^E$, where $\mathscr{F}_i = \tilde{R}(\boldsymbol{y})$, in case the human gives an absolute evaluation (denoted by $\tilde{R}$) on parametrization $\boldsymbol{\omega}_i$ in context $\boldsymbol{s}_i$, $\boldsymbol{y} = [\boldsymbol{s}^T, \boldsymbol{\omega}^T]^T$. In case of preference feedback $\mathscr{F}_i = \boldsymbol{y}_k \succ \boldsymbol{y}_{i\neq k}$ if $\boldsymbol{y}_k$ is preferred over $\boldsymbol{y}_i$. Note that for a given sample there may exist both preference and absolute evaluation.

We define the prior distribution over the latent rewards as a Gaussian Process [23], $\hat{\boldsymbol{R}} \sim \mathscr{N}(\boldsymbol{0}, \boldsymbol{K})$, with $\boldsymbol{K}_{ij} = k(\boldsymbol{y}_i, \boldsymbol{y}_j)$. Without the loss of generality we assume $\boldsymbol{0}$ prior mean, but more informative priors can be constructed with expert knowledge. The likelihood function for preference based feedback is given by $p(\boldsymbol{y}_i \succ \boldsymbol{y}_j|\hat{\boldsymbol{R}}) = \boldsymbol{\Phi}((\hat{R}_i - \hat{R}_j)/(\sqrt{2}\sigma_p))$ [7], where $\boldsymbol{\Phi}(\cdot)$ is the c.d.f. of $\mathscr{N}(0, 1)$ and σ_p is a noise term accounting for feedback noise. For absolute feedback data we simply define the likelihood by $p(\tilde{R}_i|\hat{\boldsymbol{R}}) = \mathscr{N}(\hat{R}_i, \sigma_r^2)$, where σ_r^2 represents the variance of absolute human feedback. Finally, the posterior distribution of the latent rewards can be approximated by

$$p(\hat{\boldsymbol{R}}|\mathscr{D}) \propto \prod_{i=1}^{N} p(\boldsymbol{y}_{i,1} \succ \boldsymbol{y}_{i,2}|\hat{\boldsymbol{R}}) \prod_{j=1}^{M} p(\tilde{R}_j|\hat{R}_j, \sigma_r^2) p(\hat{\boldsymbol{R}}|\boldsymbol{0}, \boldsymbol{K}), \tag{2}$$

where we used the notation $p(\boldsymbol{y}_{i,1} \succ \boldsymbol{y}_{i,2}|\hat{\boldsymbol{R}})$ to highlight that $\mathscr{F}_i$ is a preference feedback comparing $\boldsymbol{y}_{i,1}$ to $\boldsymbol{y}_{i,2}$. For finding the optimal latent rewards, we minimize

$$J(\hat{\boldsymbol{R}}) = -\sum_{i=1}^{N} \log \boldsymbol{\Phi}(z_i) + \frac{\sigma_r^{-2}}{2} \sum_{j=1}^{M} (\tilde{R}_j - \hat{R}_j)^2 + \hat{\boldsymbol{R}}^T \boldsymbol{K}^{-1} \hat{\boldsymbol{R}}, \tag{3}$$

with $z_i = (\hat{R}(\boldsymbol{y}_{i,1}) - \hat{R}(\boldsymbol{y}_{i,2}))/(\sqrt{2}\sigma_p)$. It was shown in [7] that minimizing J w.r.t. $\hat{\boldsymbol{R}}$ is a convex problem in case there is only preference based feedback ($M = 0$). However, it easy to see that the Hessian of $J(\hat{\boldsymbol{R}})$ will only be augmented with non-negative elements in the diagonal in case $M > 0$, which will leave the Hessian positive semi-definite and the problem convex. Optimizing the hyper-parameters of the kernel function $\boldsymbol{\theta}$ and the noise terms can be evaluated by maximizing the evidence $p(\mathscr{D}|\boldsymbol{\theta}, \sigma_p, \sigma_r)$. While the evidence cannot be given in a closed form, we can estimate it by Laplace approximation.

It is interesting to note that in case there is only preference feedback, that is, $M = 0,\ N > 0$, we obtain the exact same algorithm as in [7]. In the other extreme, in case there is only absolute feedback ($M > 0,\ N = 0$) we get Gaussian Process regression, which provides a closed form solution for $p(\hat{\boldsymbol{R}})$. Overall, our extension provides an opportunity to mix preference and absolute feedback in a unified Bayesian framework.

Also note that after obtaining $p(\hat{\boldsymbol{R}})$ we can use Bayesian linear regression to query the expected reward R^* of unseen samples $\boldsymbol{y}^*$ [7, 23]. We can use the resulting generative model of the reward to query the reward for a large number of samples from the current control distribution $\boldsymbol{y} \sim \mu(\boldsymbol{s})\pi(\boldsymbol{\omega}|\boldsymbol{s})$, without the need for real experimental evaluation. Such a data-efficient model-based approach has been demonstrated to reduce the required number of experiments up to two orders of magnitude [10, 19].

3.4 Contextual Relative Entropy Policy Search

To update the policy $\pi(\boldsymbol{\omega}|\boldsymbol{s})$, we rely on the contextual extension of Relative Entropy Policy Search [11, 19], or C-REPS. The intuition of C-REPS is to maximize the expected reward over the joint context-control parameter distribution, while staying close to the observed data to balance out exploration and experience loss. C-REPS uses an information theoretic approach, where the relative entropy between consecutive parameter distributions is bounded $\int_{\boldsymbol{s},\boldsymbol{\omega}} p(\boldsymbol{s}, \boldsymbol{\omega}) \log \frac{p(\boldsymbol{s},\boldsymbol{\omega})}{q(\boldsymbol{s},\boldsymbol{\omega})} d\boldsymbol{s} d\boldsymbol{\omega} \leq \varepsilon$, where $p(\boldsymbol{s}, \boldsymbol{\omega})$ and $q(\boldsymbol{s}, \boldsymbol{\omega})$ represent the updated and the previously used context-parameter distributions. The parameter $\varepsilon \in \mathbb{R}^+$ is the upper bound of the relative entropy. The emerging constrained optimization problem can be solved by the Lagrange multiplier method (see e.g. [18]). The closed form solution for the new distribution is given by $p(\boldsymbol{s}, \boldsymbol{\omega}) \propto q(\boldsymbol{s}, \boldsymbol{\omega}) \exp\left((R(\boldsymbol{\omega}, \boldsymbol{s}) - V(\boldsymbol{s}))/\eta\right)$. Here, $V(\boldsymbol{s})$ is a context dependent baseline, while η and $\boldsymbol{\theta}$ are Lagrangian parameters. The baseline is linear in some context features and it is parametrized by $\boldsymbol{\theta}$. To update the policy we use the computed probabilities $p(\boldsymbol{s}, \boldsymbol{\omega})$ as sample weights and perform a maximum likelihood estimation of the policy model parameters.

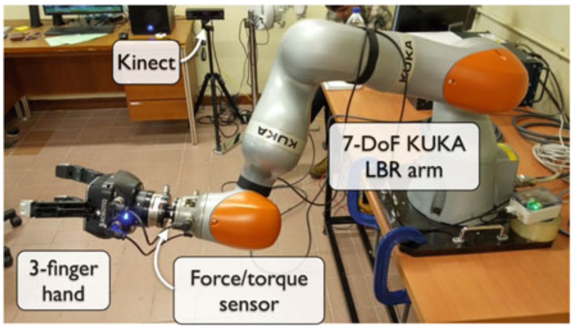


Fig. 3 Robot setup for experiments. We use the 7-DoF KUKA LBR arm with the 3-finger Robotiq robot hand. We use Kinect to track the human hand motion

4 Experiments

For the handover experiment we use the 7-DoF KUKA LBR arm (Fig. 3). For the robot hand we use the Robotiq 3-finger hand. The fingers are position controlled, but the maximum grip force can be indirectly adjusted by limiting the finger currents. In order for accurate measurement of external forces and torques, a wrist mounted force/torque sensor is installed.

4.1 Experimental Setup

An experiment is executed as follows. First, a 1.5l water bottle is placed at a fixed location, which the robot is programmed to pick up. Subsequently, the robot moves the bottle to a predefined position. At this point we enable compliant arm control and we use a Kinect sensor (Fig. 3) to track the hand of the human. Subsequently, the human moves towards the robot to take the bottle. While approaching the robot, we use the Kinect data to estimate the hand velocity $\boldsymbol{s}$ of the human, which we assume to be constant during the experiment. We only use data when the human is relatively far (above 1 m) from the robot to avoid occlusion. After the context variable is estimated the robot sets its parameter by drawing a controller parametrization $\boldsymbol{\omega} \sim \pi(\boldsymbol{\omega}|\boldsymbol{s})$. Subsequently, the robot and the human make physical contact and the handover takes place. Finally, the human evaluates the robot performance (preference or absolute evaluation on a 1–10 scale, where 1 is worst 10 is best) and walks away such that the next experiment may begin.

We presented the pseudo code of our learning algorithm in Table 1. As input to the algorithm we have to provide the initial policy $\pi(\boldsymbol{\omega}|\boldsymbol{s})$, and several other parameters. We use a Gaussian distribution to represent the policy $\pi(\boldsymbol{\omega}|\boldsymbol{s}) = \mathcal{N}(\boldsymbol{\omega}|\boldsymbol{a} + \boldsymbol{A}\boldsymbol{s}, \boldsymbol{\Sigma})$. In the beginning of the learning we set $\boldsymbol{A} = \boldsymbol{0}$, that is, the robot uses the same controller distribution over all possible context values. During learning all the parameters $(\boldsymbol{a}, \boldsymbol{A}, \boldsymbol{\Sigma})$ of the policy will be tuned according to the C-REPS update rule.

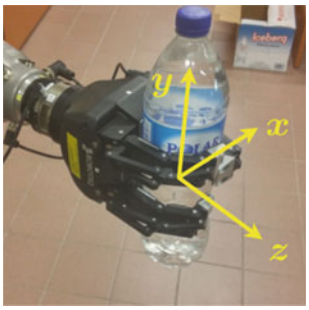


Fig. 4 The robot hand frame orientation

The initial policy mean $\boldsymbol{a}$ and the diagonal elements of the covariance matrix $\boldsymbol{\Sigma}$ are set as follows. For the rotational stiffness we set 2.75 Nm/rad mean and 0.5^2 variance. For the translational stiffness parameters we chose $275,\ 450,\ 275$ N/m in x, y, and z direction in the hand frame (Fig. 4). The variances are 50^2, 75^2, and 50^2 respectively. For the finger control slope parameter we chose 2.5 1/N with a variance of 0.5^2. This provides a firm grip of the water bottle. The robot will not move the fingers until the force generated by the human hand reaches half the weight of the bottle. With a slope parameter of 0 the robot exerts a minimal grip force that can still support the bottle. With a slope value above 5 the robot only releases the bottle if the human can support $1.2\times$ the object weight. Thus, we can avoid dropping the object, even with the initial policy. Finally as mean we set 200 and 600 mm as minimal and maximal trajectory tracking control distance. As variances we chose 50^2 and 150^2. The parameters are therefore initialized as $\boldsymbol{a} = (2.75,\ 275,\ 450,\ 275,\ 2.5,\ 200,\ 600)^T$, $\boldsymbol{A} = \boldsymbol{0}$ and $\boldsymbol{\Sigma} = \text{diag}(0.5^2,\ 50^2,\ 75^2,\ 50^2,\ 0.5^2,\ 50^2,\ 150^2)$.

For the C-REPS learning algorithm in Table 1 we chose $\varepsilon = 0.75$ and we updated the policy after evaluating $L = 10$ human-robot handover experiments. However, before the first policy update we used $L = 40$ handover experiments, such that we have a reliable estimation of the latent rewards. Before each policy update we estimate the latent rewards for all the previously seen experiments $\{\boldsymbol{\omega}_i, \boldsymbol{s}_i, \mathscr{F}_i\}_{i=1}^{E}$. Here, E represents the total number of observed samples. Note, that E is increased by the amount of latest experiments L before each policy update. Therefore, E represents how much experimental evaluation, or information we used to reach a certain level of performance. After estimating the latent rewards we use the resulting generative reward model to evaluate $Q = 500$ *artificial* context-control parameter pairs drawn from $\hat{\mu}(\boldsymbol{s})\pi(\boldsymbol{\omega}|\boldsymbol{s})$. We used these artificial samples to update the policy. This way we got a highly data-efficient algorithm, similar to the one in [19]. After the policy is updated, we start a new loop and evaluate L new experiment. We not only use this information to update our dictionary to estimate latent rewards, but also to estimate the performance of the current policy. The performance of the policy is measured by the expected latent reward of the newly evaluated L experiments. We expect the performance measure to increase with the amount of information E and policy updates. After updating the policy H times (Table 1) we terminate the learning.

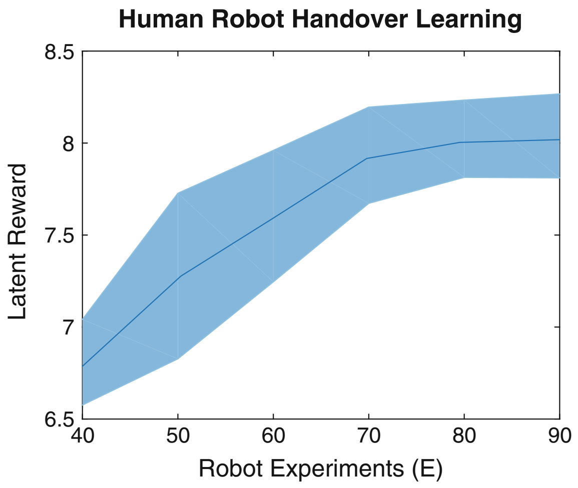


Fig. 5 The expected latent reward mean and standard deviation over 5 independent learning trials

4.2 *Results*

As the learning algorithm uses randomly sampled data for policy updates and noisy human feedback, the learned policy and its performance may vary. In order to measure the consistency of the learning progress we repeated the complete learning trial several times. A trial means evaluating the learning algorithm starting with the initial policy and with an empty dictionary, $E = 0$, but using the same parameters for L and ε. We evaluated 5 learning trials and recorded the expected performance of the robot before each policy update. The expected learning performance over 5 trials with 95% confidence bounds against the amount of real robot experiments E used for policy update is shown in Fig. 5. We can see that learning indeed improved the performance of the initial policy, which has an expected value of 6.8. Over the learning trials, we noticed that the human mostly gave absolute feedback for very good or bad solutions. This is expected as humans can confidently say if a handover skill feels close to that of a human, or if it does something unnatural (e.g., not releasing the object). By the end of the learning, the expected latent reward rose to the region of 8. Note, that the variance of the learning performance over different trials not only depends on the stochastic learning approach, but also on noisy human feedback. Thus we can conclude that the learning indeed improved the expected latent reward of the policy, but how did the policy and the experiments change with the learning?

The learned policy. We first discuss the mean value a of the learned policy and then we show how the policy generalizes to more dynamic tasks. Over several learning trials we observed that a high quality policy provides a lower rotational stiffness compared to the hand-tuned initial policy. We observed that on expectation the learned rotational stiffness is 1.29 Nm/rad, which is lower than the initial 2.75. This helped the robot to quickly orient the object with the human hand upon physical contact. We observed similar behavior in the translational stiffness values in the $x - z$

directions (see Fig. 4). The learned values were almost 100 N/m lower compared to the initial values. This helps the robot to become more compliant in horizontal motions. Interestingly, the learned stiffness in y direction became slightly higher (474 N/m) compared to its initial value. During physical interaction the forces acting along the y-axis are mostly responsible for supporting the weight of the object. With a higher stiffness value interaction times became lower and also helped avoiding situations where the robot did not release the object. The learned slope parameter of the finger controller became more conservative (3.63 1/N). This prevented any finger movement until the human force reached at least $0.8\times$ the weight of the object. Finally, the learned minimal and maximal tracking distance on expectation became 269 and 541 mm respectively.

The policy generalizes the controller parametrization with mean $\boldsymbol{a} + \boldsymbol{A}\boldsymbol{s}$. We discussed above how $\boldsymbol{a}$ changed on expectation after the learning. We now turn our attention to $\boldsymbol{A}$ and show how generalization to more dynamic task happens. We typically executed experiments with hand speed between 0.1 and 1 m/s. We observed that on expectation the rotational stiffness values were lowered for more dynamical tasks ($\boldsymbol{s} = 1$ m/s) with -0.31 Nm/rad. This helped the robot to orient with the human hand quicker. Interestingly, we observed that the stiffness in x direction is slightly increased with 56 N/m. However, the stiffness in y direction is dramatically decreased with -281 N/m. This reduces forces acting on the human significantly during faster physical interaction. The stiffness in z direction is decreased with -10 N/m, which is just a minor change. Interestingly, the slope parameter of the robot finger controller increases with 0.6 1/N, which leads to an even more conservative finger control. Finally, we observed that on expectation the minimal hand tracking distance is increased by 46 mm and the maximal distance remains almost the same with an additional 9 mm. A visual representation of the learned parameters against the context

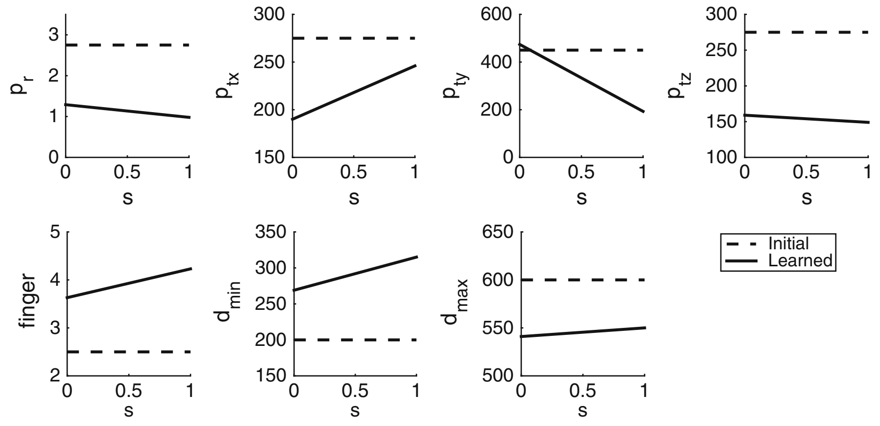


Fig. 6 The initial and the learned policy parameters against the context value. *Top row*, from *left* to *right*: the rotational stiffness, translational stiffness in the x-y-z direction. *Bottom row*, from *left* to *right*: finger control slope, minimal and maximal visual hand tracking distance

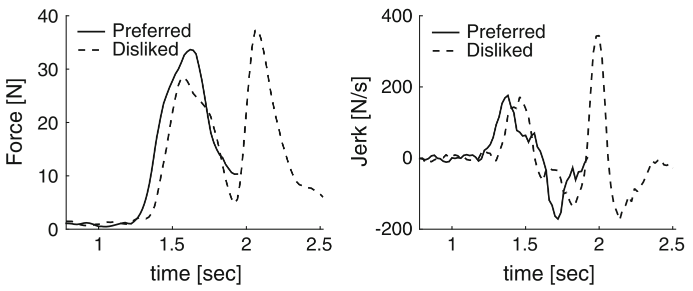


Fig. 7 Two example of experimental results of the forces acting between the human and the robot during physical interaction. The forces are plotted starting right before the physical interaction until the handover is finished

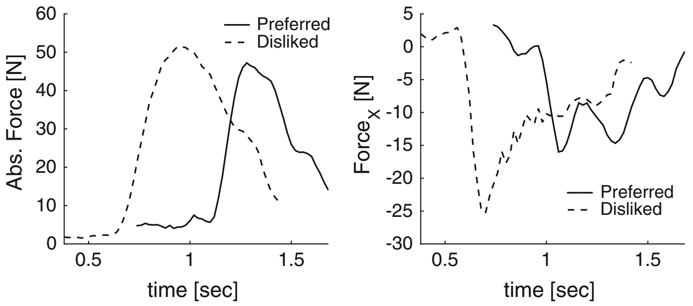


Fig. 8 Two example of experimental results in dynamic handover situations. The forces are plotted starting right before the physical interaction until the handover is finished

value is shown in Fig. 6. In the following, we will analyze some static and dynamic handover experiments to give more insight why humans prefer the learned policy as opposed to the initial hand-tuned controller.

Human preferences for static handovers. For static handover tasks we observed that a robust and quick finger control was always preferred and highly rated. In Fig. 7 we can see the forces and jerks of two typical static handover solutions. The weight of the bottle was around 15 N. We can see that the preferred solution always maintained a low jerk and forces remained limited. Moreover, a successful handover happens relatively fast. In our experiments we observed that a high quality solution happens within 0.6 s and no faster than 0.4 s. Similar results have been reported in human-human object transfer experiments [5]. Typically disliked parameterizations have low translational stiffness and a stiff finger control, resulting in the robot not releasing the object quick enough, which is considered a failure. These experiments typically lasted for 1–2 s until the bottle was released.

Human preferences for dynamic handovers. In dynamic handover situations contact forces and jerks were significantly higher compared to the static case (Fig. 8). A typical preferred dynamic handover controller has lower rotational and translational stiffness, and a more firm finger controller. In our experiments the human always came from one direction while taking the bottle from the robot. In the robot hand frame this was the x-direction. As we can see, a preferred controller achieves a significantly lower contact force and jerk in this direction. We noticed that a physical contact time in a dynamic handover scenario is around 0.3–0.6 s. Based on the latent rewards, we noticed that there is a strong preference towards faster handovers, as opposed to the static case, where we did not observe such strong correlation in handovers within 0.6 s. Interestingly, we noticed that humans preferred stiffer finger controllers in dynamic handovers. We assume that this helps a robust transfer of the object from giver to taker. In a dynamic handover situation vision might not provide enough feedback about the handover situation during physical contact, and thus, an excess of grip force would be necessary to ensure the robust transfer and to compensate for inaccurate position control.

Video footage of some typical experiments before and after the learning is available at www.youtube.com/watch?v=2OAnyfph3bQ.

By analyzing these experiments we can see that the learned policy indeed provides a controller parametrization that decreases handover time, reduces forces and jerks acting on the human over a wide variety of dynamic situations. While the initial policy provides a reasonable performance in less dynamic experiments, learning and generalization significantly improves the performance of the policy. Based on our observations, for static handovers a fast and smooth finger control was necessary for success, while in dynamic handover situation higher compliance and a firm finger control were preferred.

5 Discussion

This paper presents an algorithm for learning dynamic robot-to-human object handover from human feedback. The algorithm learns a latent reward function from both absolute and preference feedback, and integrates reward learning with contextual policy search. Experiments show that the robot adapts to the dynamics of human motion and learns to hand over a water bottle successfully, even in highly dynamic situations.

The current work has several limitations. First, it is evaluated on a single object and a small number of people. We plan to generalize the learning algorithm to adapt over human preferences and object characteristics. While contextual policy search works well for adapting over handover dynamics, object characteristics exhibit much greater variability and may pose greater challenge. Second, our handover policy also does not consider human response during the handover or its change over time. We want to model key features of human response and exploit it for effective and fluid handover. For both, combining model-free learning and model-based planning seems a fruitful direction for exploration.

Acknowledgements This research was supported in part an A*STAR Industrial Robotics Program grant (R-252-506-001-305) and a SMART Phase-2 Pilot grant (R-252-000-571-592).

References

1. Agah, A., Tanie, K.: Human interaction with a service robot: mobile-manipulator handing over an object to a human. In: Proceedings of the IEEE International Conference on Robotics and Automation (1997)
2. Ben Amor, H., Neumann, G., Kamthe, S., Kroemer, O., Peters, J.: Interaction primitives for human-robot cooperation tasks. In: Proceedings of the IEEE International Conference on Robotics and Automation (2014)
3. Bruno, S., Khatib, O. (eds.): Handbook of Robotics. Springer, Berlin (2008)
4. Cakmak, M., Srinivasa, S., Lee, M., Forlizzi, J., Kiesler, S.: Human preferences for robot-human hand-over configurations. In: Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (2011)
5. Chan, W., Parker, C., Van der Loos, H., Croft, E.: A human-inspired object handover controller. Int. J. Robot. Res. **32**(8), 971–983 (2013)
6. Chan, W.P., Kumagai, I., Nozawa, S., Kakiuchi, Y., Okada, K., Inaba, M.: Implementation of a robot-human object handover controller on a compliant underactuated hand using joint position error measurements for grip force and load force estimations. In: Proceedings of the IEEE International Conference on Robotics and Automation (2014)
7. Chu, W., Ghahramani, Z.: Preference learning with Gaussian processes. In: Proceedings of the International Conference on Machine Learning (2005)
8. da Silva, B., Konidaris, G., Barto, A.: Learning parameterized skills. In: Proceedings of the International Conference on Machine Learning (2012)
9. Daniel, C., Neumann, G., Peters, J.: Hierarchical relative entropy policy search. In: AISTATS (2012)
10. Daniel, C., Viering, M., Metz, J., Kroemer, O., Peters, J.: Active reward learning. In: Proceedings of the Robotics: Science and Systems (2014)
11. Deisenroth, M.P., Neumann, G., Peters, J.: A survey on policy search for robotics. Found. Trends Robot. **2**(1–2), 1–142 (2013)
12. Dragan, A., Srinivasa, S.: Generating legible motion. In: Proceedings of the Robotics: Science and Systems (2013)
13. Grigore, E.C., Eder, K., Pipe, A.G., Melhuish, C., Leonards, U.: Joint action understanding improves robot-to-human object handover. In: Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems, pp. 4622–4629. IEEE (2013)
14. Huang, C.-M., Cakmak, M., Mutlu, B.: Adaptive coordination strategies for human-robot handovers. In: Proceedings of the Robotics: Science and Systems (2015)
15. Huber, M., Kupferberg, A., Lenz, C., Knoll, A., Brandt, T., Glasauer, S.: Spatiotemporal movement planning and rapid adaptation for manual interaction. PLoS One (2013)
16. Ijspeert, A.J., Schaal, S.: Learning attractor landscapes for learning motor primitives. In: Advances in Neural Information Processing Systems (2003)
17. Jain, A., Wojcik, B., Joachims, T., Saxena, A.: Learning trajectory preferences for manipulators via iterative improvement. In: Advances in Neural Information Processing Systems (2013)
18. Kupcsik, A., Deisenroth, M., Peters, J., Ai Poh, L., Vadakkepat, V., Neumann, G.: Model-based contextual policy search for data-efficient generalization of robot skills. Artif. Intell. (2015)
19. Kupcsik, A., Deisenroth, M.P., Peters, J., Neumann, G.: Data-efficient contextual policy search for robot movement skills. In: Proceedings of the AAAI Conference on Artificial Intelligence (2013)
20. Mainprice, J., Gharbi, M., Siméon, T., Alami, R.: Sharing effort in planning human-robot handover tasks. In: Proceedings of the International Symposium on Robot and Human Interactive Communication (2012)
21. Nagata, K., Oosaki, Y., Kakikura, M., Tsukune, H.: Delivery by hand between human and robot based on fingertip force-torque information. In: Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (1998)
22. Ng, A., Russell, S.: Algorithms for inverse reinforcement learning. In: Proceedings of the International Conference on Machine Learning (2000)

23. Rasmussen, C.E., Williams, C.K.I.: Gaussian Processes for Machine Learning. The MIT Press, Cambridge (2005)
24. Ratliff, N., Silver, D., Bagnell, J.: Learning to search: functional gradient techniques for imitation learning. Auton. Robot. **27**(1), 25–53 (2009)
25. Sisbot, E., Alami, R., Siméon, T., Dautenhahn, K., Walters, M., Woods, S.: Navigation in the presence of humans. In: Proceedings of the IEEE-RAS International Conference on Humanoid Robots (2005)
26. Strabala, K., Lee, M.K., Dragan, A., Forlizzi, J., Srinivasa, S., Cakmak, M., Micelli, V.: Towards seamless human-robot handovers. J. Hum.-Robot Interact. (2013)
27. Wilson, A., Fern, A., Tadepalli, P.: A Bayesian approach for policy learning from trajectory preference queries. In: Advances in Neural Information Processing Systems (2012)
28. Wirth, C., Fürnkranz, J.: Preference-based reinforcement learning: a preliminary survey. In: Fürnkranz, J., Hüllermeier, E. (eds.) Proceedings of the ECML/PKDD Workshop on Reinforcement Learning from Generalized Feedback: Beyond Numeric Rewards (2013)

Decoding the Neural Mechanisms Underlying Locomotion Using Mathematical Models and Bio-inspired Robots: From Lamprey to Human Locomotion

Auke Jan Ijspeert

1 Introduction

Understanding animal locomotion is a complex problem because locomotion is the result of a complex interaction between multiple components [1]. At an abstract level, four different components can be distinguished: the musculoskeletal system, sensory feedback loops in the spinal cord, central pattern generators CPGs (neural circuits that can produce rhythmic patterns without receiving rhythmic inputs), and descending modulation from higher control centers (such as the basal ganglia, the cerebellum, and the motor cortex in mammals).

Robots together with computational models can serve as useful scientific tools to (i) explore the interaction of these different components, and (ii) investigate how their respective roles have changed during evolution [1–3].

During the evolution from fish to mammals, the role of CPGs appears to diminish, and those of sensory feedback and descending modulation to increase with the apparition of limbs and the transition from sprawling postures like in salamanders to upright postures of mammals (Fig. 1). The unstable upright postures and the importance of visually-guided feet placement have required a more prominent role of sensory feedback and of descending modulation. In humans, the locomotor circuits are still far from being understood, and there is still a debate whether human locomotion involves CPGs or not, and whether the controllers are mainly in the spinal cord or in the motor cortex.

In order to investigate these components and their interaction for different animals, we have developed models of locomotor circuits for robots that follow evolution and take inspiration from the lamprey, the salamander, the cat, and the human (Fig. 2). This type of research is not only interesting for understanding animal locomotion, but also for robotics as it addresses problems related to tradeoffs between feedback and

A.J. Ijspeert (✉)
Biorobotics Laboratory, EPFL - Ecole Polytechnique Fédérale de Lausanne EPFL-STI-IBI, Station 9, 1015 Lausanne, Switzerland
e-mail: auke.ijspeert@epfl.ch

A. Bicchi and W. Burgard (eds.), *Robotics Research*, Springer Proceedings in Advanced Robotics 2, DOI 10.1007/978-3-319-51532-8_11

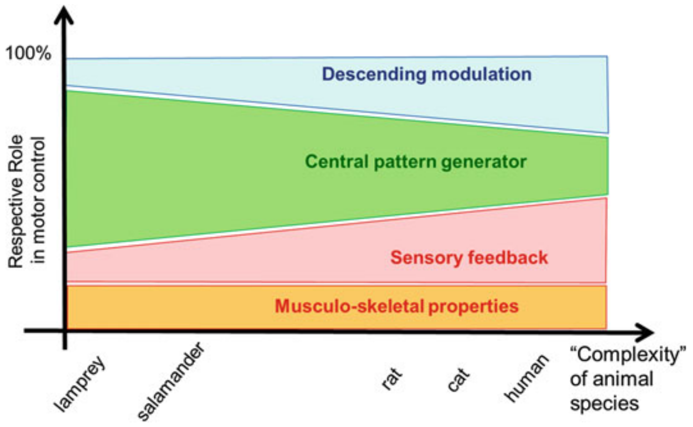


Fig. 1 Schematic view of the roles of the four components underlying vertebrate locomotion. *Note* this is a hypothetical view based on the author's review of animal locomotion control literature. Quantifying the respective roles is hardly possible, but computational models can help providing a qualitative idea of the importance of each component for different animals. The horizontal axis is somewhat subjectively called "complexity". The ordering of animal species corresponds to their appearance on an evolutionary time scale. One possible way of measuring complexity could be the number of neurons in the central and peripheral nervous systems; another could be the instability of locomotion (e.g. the upright and bipedal nature of human locomotion makes it less stable, i.e. more sensitive to perturbations, than lamprey and salamander locomotion)

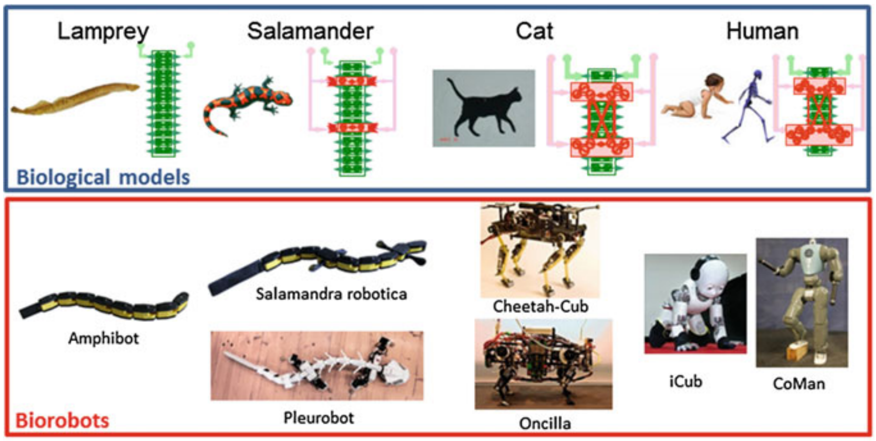


Fig. 2 Range of models and robots to investigate the evolution of locomotor circuits in vertebrates

feedforward control, robust trajectory generation, and more generally locomotion in unstructured terrain. In the next sections, the main outcomes of these studies are briefly presented as well as those of related work.

2 Lamprey Models

The lamprey is a primitive fish that swims using an anguilliform swimming gait in which a traveling wave of body undulation is propagated from head to tail. Because of its relative simplicity, it has been extensively studied by neuroscientists [4, 5]. One of the main findings is that the lamprey's spinal cord is capable of generating fictive locomotion: i.e. the ability of an isolated spinal cord in a petri dish to produce rhythmic patterns of neural activity that closely resemble those of intact locomotion. This demonstrates the existence of a locomotor central pattern generator in the spinal cord. Another main finding is the identification of the segmental circuits and neurons involved in rhythm generation.

The swimming circuit has since been modeled at several levels of abstraction from detailed neuronal models to more abstract models made of coupled nonlinear oscillators (see references in [6]). These models have contributed to a better understanding of the mechanisms of rhythm generation in segmental circuits, of generation of traveling waves, and of the role of sensory feedback.

The lamprey is also a good source of inspiration for robotics and several lamprey-like robots have been constructed [7–9]. In my laboratory, we have designed such a lamprey-like robot, Amphibot (Fig. 2, [10, 11]), to investigate the neural mechanisms of locomotion control in vertebrates, and their evolution from water to ground locomotion (see next section).

Using a CPG modeled as a chain of coupled oscillators, we could generate swimming gaits that are modulated by simple descending modulation for adjusting the speed and heading [11]. The speed and heading can be interactively adjusted with a remote control, and abrupt changes of modulation signals lead to smooth and continuous modulation of the CPG activity (because of the limit cycle behavior of the coupled oscillators). This could be done in open-loop, i.e. without sensory feedback affecting the CPG activity. These findings suggest that CPG networks could be the main component underlying lamprey locomotion (Fig. 1).

Sensory feedback is not necessary for generating periodic patterns in lamprey locomotion, but it plays a role in modulating the patterns to adapt to the environment. In simulation, it was shown by Ekeberg and colleagues that sensory feedback from stretch sensitive cells can improve swimming in non-steady state conditions, for instance when swimming against a speed barrier [12]. Experiments will be performed in a near future to test whether similar results can be replicated with the real robot. Furthermore we are also exploring how sensory feedback might contribute to adjust the traveling waves, in particular the phase lags between segments. Preliminary experiments tend to show that feedback can correct traveling waves that do not have the right phase lags (e.g. too large phase lags), i.e. that the body and the sensory feedback can act as a filter that corrects wrong open loop patterns.

3 Salamander Models

The transition from water to ground has been a key moment during evolution. The salamander is an interesting animal to investigate the changes in morphology and control that have taken place during that transition. Indeed the body plan of the salamander is very close to that found in fossils of the first terrestrial tetrapods [13].

The salamander swims using an anguilliform swimming gait like the lamprey. On ground it switches to a walking trot gait in which the trunk and tail perform and S-shaped standing wave [14]. Although the organization of the salamander locomotor circuit is less well-known than that of the lamprey, it appears to share many similarities with it [15]. In addition to have a lamprey-like CPG for its axial musculature, it has specialized oscillatory centers for the limbs, the limb CPGs. Interestingly, locomotion can be induced by electrical stimulation of a particular region in the brainstem of the salamander: this induces walking-like gaits at low stimulation, and swimming-like gaits at high stimulation [16].

We have developed Salamandra robotica, a salamander-like robot capable of swimming and walking [13, 17]. The robot allowed us to test several hypotheses about the reorganization of the locomotor circuits during the transition between water and land, and the apparition of limbs. Our main hypothesis is that the salamander has kept a lamprey-like CPG that tends to produce traveling waves for its axial musculature, and that it is extended by slower limb CPGs that impose slower walking gaits when activated. The coupled oscillator model that we developed to test those hypotheses could replicate the typical swimming and walking gaits of the salamander and dynamically switch between the two depending on the level of descending drive, like in the animal [13]. We tested several options and the fastest locomotion on ground is obtained with the same body-limb coordination as the animal [17].

Interestingly, similarly to the lamprey, the robot performs well in open-loop, i.e. with the CPG and without sensory feedback; suggesting again an important role of the CPG. Control of speed and heading is similarly obtained with 2 descending modulation signals one to the left oscillators one to the right ones. Because of the sprawling posture, accurate feet placement is not necessary and the robot does not fall over on ground while performing the walking trot gait. While demonstrating that salamander-like locomotion could be obtained in open loop, we believe that feedback plays an important role in shaping the locomotor patterns, in particular the transition between the traveling wave undulation during swimming and the standing waves during walking. In simulation, different interaction forces in water and on ground together with sensory feedback from stretch sensors could explain the different muscle activity patterns [18]. Similar tests will soon be performed with the robot.

We have recently developed a new robot, Pleurobot (Fig. 2), with 27 actuated degrees of freedom to explore rich motor skills. In particular, we are interested in investigating how more complex descending modulation can lead to more complex motor behaviors. We are exploring how several additional descending modulation pathways can be added to our spinal networks in order to perform various motor behaviors such as turning, backward stepping, moving single limbs and other non-steady-state maneuvers that the animal exhibits. Preliminary experiments show that activating differentially different parts of the CPG network is sufficient to generate a larger range of motor behaviors compared to the swimming and walking gaits that we initially investigated.

4 Cat Models

Cats have impressive locomotion skills: they can run, jump, climb trees, walk on thin branches, etc. Due to the upright posture, sensory feedback plays a bigger role than in salamander for keeping balance. Also cats are capable of complex visuomotor coordination tasks (e.g. placing feet at specific locations and manipulating objects) that require more complex descending modulation than salamander or lamprey. Neuroscientists have again demonstrated that the spinal cord circuits play a major role in generating the locomotion patterns [19]. The exact circuits are not as well-known as for the lamprey, but multiple reflex loops have been identified [20, 21]. Also several interesting neural models have been developed to investigate the underlying circuitry [22, 23].

Several impressive quadruped robots have been developed: BostonDynamics' BigDog and Cheetah, MIT cheetah, SONY's Aibo, StarlETH, and Tekken, to name a few. Among these, the Tekken robot has been used by Kimura and colleagues to explore the interaction between CPGs and reflexes [24]. They explored different manners in which CPGs and reflexes could be integrated and found out that the most robust locomotion is obtained when sensory feedback shares interneurons with the CPG, compared to feedback that is independent and does not affect the CPG. This is in agreement with phase-dependent reflexes observed in cats [25].

In my laboratory, we have designed two cat-like robots, Cheetah-Cub and Oncilla (Fig. 2). Cheetah-Cub was designed as a light-weight robot to investigate the role of visco elastic properties of the body in dynamic locomotion [26]. The legs approximately match the cat leg design and are made of 3 segments in a pantograph structure with springs. The robot can produce fast trotting gaits of almost 7 body length per seconds. Interestingly those gaits were obtained with (well-tuned) open loop patterns. Furthermore the gaits are robust against perturbations such as a step down. This demonstrates interesting self-stabilizing properties of the (robot) body, and gives some hints about how properties of the musculo-skeletal system could simplify control in vertebrate animals. This is well-aligned with the notion of "embodied intelligence" proposed by Pfeifer and colleagues [27].

Oncilla (Fig. 2) is a robot equipped with various sensors (accelerometers, gyroscopes, 3D force sensors on feet) that we developed to investigate the interplay between CPGs, reflexes, and posture control [28]. The posture control is inspired from virtual model control [29]. Two reflexes are included: a stumbling correction reflex and a leg extension reflex. In simulation, we tested different combinations of CPGs, reflexes and posture control (but without vision), and investigated how well the controllers performed in terrains with slopes, step downs, and uneven ground. As expected, the combination of CPGs, reflexes and posture control was necessary to handle the most complex terrains compared to controllers with only CPGs or only CPGs and posture control [28]. The same controllers have been successfully ported to the real robot (manuscript in preparation).

5 Human Models

The locomotor control circuits in humans have not yet been properly decoded. There is still a debate whether the control circuits include CPGs or not, and whether locomotion is mainly controlled by the motor cortex or by the spinal cord [30]. Numerical models can be very useful to test several options. An influential neuromechanical model of human locomotion was developed by Taga [31]. He demonstrated that stable locomotion could emerge from the coupling of a CPG, reflex loops and a simple mechanical model of a human body. Geyer and colleagues have since then demonstrated that human-like gaits could be obtained with neuromechanical models that are purely sensory-driven, i.e. without CPGs [32]. The gaits produced are strikingly human-like both in terms of kinematics and dynamics. This raises the question about what could be the added value of including a CPG in such a network. With colleagues in my laboratory, we have the following three hypotheses about the usefulness of adding CPGs to Geyer's sensory driven network: (1) the addition of a CPG can simplify the control of speed, (2) it can make the circuit more robust against sensory noise, and (3) it can make the circuit more robust against sensory lesions.

We tested the first hypothesis by replicating Geyer's model and investigating different options of how a CPG could be added to it [33]. We found that adding a CPG to the circuits controlling hip motion could lead to simple and robust speed control. In Geyer's original model, changing speed of locomotion involves changing multiple reflex gains in a rather complex manner. With a CPG, simply modulating the intrinsic frequency of the oscillators allows adjusting the frequency of stepping and hence the speed of walking. This tends to confirm our first hypothesis. We developed a similar controller for a compliant biped robot COMAN (Fig. 2) in simulation, with successful control of speed and step size [34]. Tests are currently under way to test the controller on the real robot.

6 Discussion

The projects presented in this article suggest that the respective roles of the musculoskeletal system, sensory feedback, CPGs, and descending modulation have changed during the evolution from aquatic to terrestrial locomotion. In lamprey and salamander, robust locomotion can be obtained by relying heavily on CPGs. Because the type of locomotion is relatively simple and because posture control is less critical than in mammals, the feedforward and open loop patterns of CPGs can by themselves produce quite robust locomotion. Furthermore the activity of the CPGs can be modulated by low dimensional descending signals to dynamically change the speed, heading and (in the salamander) the type of gait.

In the cat and in human locomotion, appear to play a much more important role, and, as demonstrated by Geyer and colleagues, locomotion could in principle be generated without CPGs. Our work tends to suggest that CPGs are still useful, but in a different manner than in the lamprey or the salamander. Instead of generating complete patterns for all degrees of freedom (DOFs), one possibility, as explored with our human locomotion study, could be that CPGs influence only a subset of DOFs (the more proximal ones like the hips) in order to help control the speed (by influencing the frequency of stepping). Furthermore, we predict that the inclusion of CPGs can help handling sensory noise and sensory lesions (cf Hypotheses 2 and 3 above) but this remains to be studied. Note that, using a pendulum like a test case, Art Kuo proposed the interesting idea that a CPG serves as a state estimator (rather than a controller) and that it acts "as an internal model of limb motion that predicts the state of the limb" [35]. It remains to be studied how this idea could extend to a complete body.

Overall, it is becoming clearer that spinal cord circuits can do (much) more than produce stereotyped locomotion (as has sometimes been believed in the past). More modeling work remains however to be done to decode how they work, in particular related to rich motor skills, non-steady state behavior, and multimodal locomotion. Also the role of properties of the musculoskeletal system in simplifying control should not be underestimated. It is difficult to assess whether this role has changed during evolution (and hence, it was kept constant in Fig. 1), but as illustrated in our work on Cheetah-Cub [26], well-tuned mechanical properties can certainly simplify control. Robotics can play a key role in decoding this type of embodied intelligence [36].

The neuroscientific research presented in this article can make several types of contributions to robotics. First of all, it demonstrates that robotics can not only benefit from biology, but that it can also contribute something in return with the design of robots that are used as scientific tools in animal motor control studies. Second, the type of models presented here can become interesting robot controllers: they are computationally cheap, are well suited for distributed implementation (e.g. different oscillators running on different microcontrollers), can be used on compliant robots with interesting passive dynamics, are well suited to be used with learning algorithms, are robust against perturbations, and can handle complex unstructured terrains.

Similarly to animals that have conquered many different ecological niches, the approaches could contribute to design controllers for field robots that have to handle complex unstructured outdoor terrains, with applications in search-and-rescue, transport, pollution monitoring, inspection, and others.

7 Conclusion

This article presented several projects in my laboratory concerning the decoding of locomotion control circuits in vertebrate animals, and the changes that they have undergone during the transition from swimming to ground locomotion. Using various robots and numerical models, our studies tend to show that locomotion in lamprey and salamander can be obtained by relying extensively on feedforward signals from CPGs, while locomotion in cat and human relies more on sensory feedback (e.g. for posture control) and on descending modulation. For all animals, the viscoelastic properties of the body should not be underestimated since they might markedly simplify control. We envision multiple future studies involving robots and numerical models to further explore the fascinating locomotor abilities of animals, and to contribute new control approaches in robotics.

Acknowledgements Swiss National Science Foundation (project CR23I2_140714), the Swiss National Center of Competence in Research in Robotics, the European Commission (projects Walkman FP7-ICT 611832, Symbitron FP7-ICT 661626, and Cogimon H2020 ICT-23-2014 644727), and the Envirobot project funded by the Swiss NanoTera program.

References

1. Ijspeert, A.J.: Biorobotics: using robots to emulate and investigate agile locomotion. Science **346**(6206), 196–203 (2014)
2. Floreano, D., Ijspeert, A.J., Schaal, S.: Robotics and neuroscience. Curr. Biol. **24**(18), R910–R920 (2014)
3. Iida, F., Ijspeert, A.J.: Biologically inspired robotics. Handbook of Robotics. Springer. In press
4. Grillner, S.: The motor infrastructure: from ion channels to neuronal networks. Nat. Rev. Neurosci. **4**(7), 573–586 (2003)
5. Grillner, S., Deliagina, T., El Manira, A., Hill, R.H., Orlovsky, G.N., Wallén, P., Ekeberg, Ö., Lansner, A.: Neural networks that co-ordinate locomotion and body orientation in lamprey. Trends Neurosci. **18**(6), 270–279 (1995)
6. Ijspeert, A.J.: Central pattern generators for locomotion control in animals and robots: a review. Neural Netw. **21**(4), 642–653 (2008)
7. Stefanini, C., Orofino, S., Manfredi, L., Mintchev, S., Marrazza, S., Assaf, T., Capantini, L., Sinibaldi, E., Grillner, S., Wallén, P., Dario, P.: A novel autonomous, bioinspired swimming robot developed by neuroscientists and bioengineers. Bioinspir. Biomim. **7**(2), 025001 (2012)
8. Wilbur, C., Vorus, W., Cao, Y., Currie, S.: A lamprey-based undulatory vehicle. Neurotechnology for Biomimetic Robots. MIT Press, Cambridge (2002)
9. Leftwich, M.C., Tytell, E.D., Cohen, A.H., Smits, A.J.: Wake structures behind a swimming robotic lamprey with a passively flexible tail. J. Exp. Biol. **215**(3), 416–425 (2012)
10. Crespi, A., Badertscher, A., Guignard, A., Ijspeert, A.J.: AmphiBot I: an amphibious snake-like robot. Robot. Auton. Syst. **50**(4), 163–175 (2005)
11. Crespi, A., Ijspeert, A.J.: Online optimization of swimming and crawling in an amphibious snake robot. IEEE Trans. Robot. **24**(1), 75–87 (2008)
12. Ekeberg, Ö., Grillner, S., Lansner, A.: The neural control of fish swimming studied through numerical simulations. Adapt. Behav. **3**(4), 363–384 (1995)
13. Ijspeert, A.J., Crespi, A., Ryczko, D., Cabelguen, J.-M.: From swimming to walking with a salamander robot driven by a spinal cord model. Science **315**(5817), 1416–1420 (2007)

14. Karakasiliotis, K., Schilling, N., Cabelguen, J.-M., Ijspeert, A.J.: Where are we in understanding salamander locomotion: biological and robotic perspectives on kinematics. Biol. Cybern. **107**(5), 529–544 (2012)
15. Chevallier, S., Jan Ijspeert, A., Ryczko, D., Nagy, F., Cabelguen, J.-M.: Organisation of the spinal central pattern generators for locomotion in the salamander: biology and modelling. Brain Res. Rev. **57**(1), 147–161 (2008)
16. Cabelguen, J.-M., Bourcier-Lucas, C., Dubuc, R.: Bimodal locomotion elicited by electrical stimulation of the midbrain in the salamander Notophthalmus viridescens. J. Neurosci. **23**(6), 2434–2439 (2003)
17. Crespi, A., Karakasiliotis, K., Guignard, A., Ijspeert, A.J.: Salamandra robotica II: an amphibious robot to study salamander-like swimming and walking gaits. IEEE Trans. Robot. **29**(2), 308–320 (2013)
18. Ijspeert, A.J., Crespi, A., Cabelguen, J.-M.: Simulation and robotics studies of salamander locomotion. Neuroinformatics **3**(3), 171–195 (2005)
19. Whelan, P.J.: Control of locomotion in the decebrate cat. Prog. Neurobiol. **49**, 481–515 (1996)
20. Pearson, K.G.: Proprioceptive regulation of locomotion. Curr. Opin. Neurobiol. **5**(6), 786–791 (1995)
21. Frigon, A., Rossignol, S.: Experiments and models of sensorimotor interactions during locomotion. Biol. Cybern. **95**(6), 607–627 (2006)
22. Ekeberg, Ö., Pearson, K.: Computer simulation of stepping in the hind legs of the cat: an examination of mechanisms regulating the stance-to-swing transition. J. Neurophysiol. **94**, 4256–4268 (2005)
23. Rybak, I.A., Stecina, L., Shevtsova, N.A., McCrea, D.A.: Modelling spinal circuitry involved in locomotor pattern generation: insights from the effects of afferent stimulation. J. Physiol. Lond. **577**, 641–658 (2006)
24. Fukuoka, Y., Kimura, H., Cohen, A.H.: Adaptive Dynamic Walking of a Quadruped Robot on Irregular Terrain Based on Biological Concepts. Int. J. Robot. Res. **3–4**, 187–202 (2003)
25. Pearson, K.G.: Generating the walking gait: role of sensory feedback. In: D.G.S., Wiesendanger, M., Mori, S. (eds.) Progress in Brain Research, vol. 143, pp. 123–129. Elsevier, Amsterdam (2004)
26. Spröwitz, A., Tuleu, A., Vespignani, M., Ajallooeian, M., Badri, E., Ijspeert, A.J.: Towards dynamic trot gait locomotion: design, control, and experiments with cheetah-cub, a compliant quadruped robot. Int. J. Robot. Res. **32**(8), 932–950 (2013)
27. Pfeifer, R., Bongard, J., Grand, S.: How the Body Shapes the Way We Think: A New View of Intelligence. MIT press, Cambridge (2007)
28. Ajallooeian, M., Gay, S., Tuleu, A., Sprowitz, A., Ijspeert, A.: Modular control of limit cycle locomotion over unperceived rough terrain. Presented at the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (2013)
29. Pratt, J., Chew, C.M., Torres, A., Dilworth, P., Pratt, G.: Virtual model control: an intuitive approach for bipedal locomotion. Int. J. Robot. Res. **20**(2), 129–143 (2001)
30. MacKay-Lyons, M.: Central pattern generation of locomotion: a review of the evidence. Phys. Ther. **82**(1), 69–83 (2002)
31. Taga, G., Yamaguchi, Y., Shimizu, H.: Self-organized control of bipedal locomotion by neural oscillators in unpredictable environment. Biol. Cybern. **65**(3), 147–159 (1991)
32. Geyer, H., Herr, H.: A muscle-reflex model that encodes principles of legged mechanics produces human walking dynamics and muscle activities. IEEE Trans. Neural Syst. Rehabil. Eng. **18**(3), 263–273 (2010)
33. Dzeladini, F., van den Kieboom, J., Ijspeert, A.: The contribution of a central pattern generator in a reflex-based neuromuscular model. Front. Hum. Neurosci. **8**, 371 (2014)
34. van der Noot, N., Ijspeert, A.J., Ronsse, R.: Biped gait controller for large speed variations, combining reflexes and a central pattern generator in a neuromuscular model. Presented at the IEEE International Conference on Robotics and Automation (ICRA 2015), Seattle, Washington, USA (2015)

35. Kuo, A.: The relative roles of feedforward and feedback in the control of rhythmic movements. Motor Control **6**, 129–145 (2002)
36. Pfeifer, R., Lungarella, M., Iida, F.: Self-organization, embodiment, and biologically inspired robotics. Science **318**(5853), 1088–1093 (2007)

Towards Real-Time SMA Control for a Neurosurgical Robot: MINIR-II

Shing Shin Cheng, Yeongjin Kim and Jaydev P. Desai

1 Introduction

Having a 5-year mortality rate of 66.5%, brain tumor was ranked 6^{th} in the list of cancers with the highest mortality rate in the United States in 2010 [1]. Neurosurgeons and medical researchers are continuously looking for ways to improve the survival rate of brain cancer patients. Minimally invasive open surgery is often recommended for tumors that are accessible and localized [2]. We envision an intraoperative magnetic resonance imaging (MRI)-guided robotic neurosurgery in which a flexible continuum robot, that is controlled by the neurosurgeons in the MRI control room, performs the surgical resection of brain tumor. In this way, we can potentially overcome challenges such as unsteady hands and tumors being out of the surgeon's line of sight.

We propose a flexible spring-based continuum robot as one of the prototypes for the minimally invasive intracranial robot (MINIR-II), that can be dexterously maneuvered in constrained and highly risky environment to avoid critical regions in the brain. Continuum robots have been explored extensively for use in the medical domain in the past decade. Examples in the literature include the tendon-driven robots [3], the pre-curved concentric tube robots [4], and a spring backbone-based robot [5]. Shape memory alloy (SMA) springs have been chosen as the actuators for this robot due to its MRI-compatibility and affordable cost [6].

S.S. Cheng (✉) · J.P. Desai
Wallace H. Coulter Department of Biomedical Engineering,
Georgia Institute of Technology, Atlanta, GA 30332, USA
e-mail: chengss90@gmail.com

J.P. Desai
e-mail: jaydev@gatech.edu

Y. Kim
Department of Mechanical Engineering,
Incheon National University, Yeonsu-gu, Incheon, Republic of Korea
e-mail: ykim@inu.ac.kr

A. Bicchi and W. Burgard (eds.), *Robotics Research*, Springer Proceedings in Advanced Robotics 2, DOI 10.1007/978-3-319-51532-8_12

SMA is a smart material that responds to temperature change. Since we use SMA tension spring in our setup, the spring contracts upon heating and relaxes upon cooling. At relatively high temperatures (between austenite start and austenite finish temperature), SMA transforms into the austenite phase which has high stiffness. At relatively low temperatures (between martensite start and martensite finish temperature), the SMA transforms into the martensite phase, that has low stiffness. It is a high power density actuator and thus can be a compact and light actuator with significant force output. SMA spring also generates significant displacement within a limited real estate [6–8].

One of the major drawbacks of SMA is its low actuation rate, limited by its cooling and heating rate [9, 10]. A neurosurgery is normally performed at a generally slow pace but requires varying of electrocauterization speed in order to remove tumor as completely as possible [11]. Several neurosurgical robots, including the NeuroArm and the ROBOCAST system, have tip speeds between 0.5 and 2 mm/s [12, 13]. Therefore, we want to devise a cooling strategy to improve the cooling rate and thus actuation bandwidth of SMA springs so that sufficiently high operation bandwidth is available for the neurosurgeons.

Several cooling strategies have been attempted on SMA wires that are used in a wide range of applications. Cooling by a fan operated at different speeds is an effective method for SMAs that are not required to be selectively cooled [14]. Heat sink has also been researched for improving SMA cooling rate. For example, an SMA wire was inserted into a metal tube that acts as a heat sink and silicone grease was stuffed into the space between the SMA and the tube for faster heat transfer [15]. A mobile heat sink was also developed to improve the actuation speed of antagonistic SMA wires [16]. A wet actuator, inspired by human's blood circulation system, was proposed to actuate an SMA wire by passing hot and cold water through a compliant channel containing the wire [17]. To date, most research efforts have focused on cooling SMA wires. Developing a compact and efficient cooling mechanism for SMA spring, which can generate larger force and displacement per unit volume, remains challenging. Inspired by the wet actuator, we propose that an SMA spring is integrated into a flexible silicone tubing, coil by coil, to form a compact cooling module integrated SMA actuator [10]. The rest of the paper can be divided into six sections: Sect. 2 introduces the 3-segmented MINIR-II, of which the base and middle segment are actuated by four pairs of SMA springs. Section 3 contains detailed elaboration on the selection of tubing for the cooling module and the process to integrate the SMA spring and the tubing. In Sect. 4, the parameters that characterize the performance of SMA actuators in terms of their actuation bandwidth are discussed and in Sect. 5, the robotic platform with the vision-based control setup is introduced. These lead to the results and discussion section in Sect. 6, where the characterization results and motion test analyses are presented. In Sect. 7, we make some concluding remarks and present related future work.

2 Multi-joint Robot Design

We envision using a meso-scale robot that can be dexterously maneuvered to go around obstacles or critical regions to reach and remove brain tumor. Our robot has a flexible inner-spring backbone that is divided into three segments: base, middle, and end (Fig. 1). The outer spring has an outer diameter of 13.2 mm and functions as a shell to maintain the robot shape and rigidity. The robot was manufactured out of VeroWhite, which is a common plastic material used in the 3-D printer (Objet 350V, Stratasys, USA). Each segment has a disk with four holes spaced equidistantly on its periphery. The holes are the terminal points for tendons connecting the robot to the SMA spring actuators. A pair of tendons are used to actuate the robot segment in one DoF and each segment offers two DoFs in the direction of pitch and yaw. Tendon routing configurations, shown in Fig. 2, were attempted on the robot. Coupling between joints was found in the robot when tendons were routed along the periphery of the robot, as shown in Fig. 2a. We were able to resolve the coupling issue using configuration shown in Fig. 2b and therefore implemented this configuration in the eventual design of the robot.

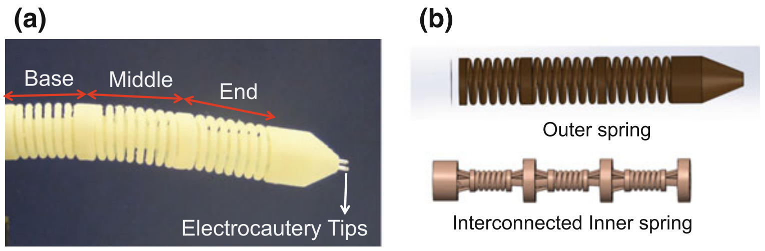


Fig. 1 **a** Spring-based MINIR-II consisting of three segments, of which the base and middle segments are actuated to achieve 4-DoF motion. (Electrocautery tips are not functional in the current robot prototype.) **b** Schematics showing outer spring and interconnected inner spring which make up the robot

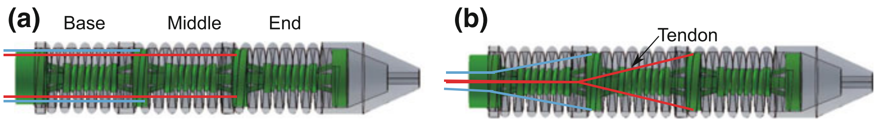


Fig. 2 Tendon routing configurations: **a** Tendons along the periphery of the robot. **b** Tendons through the center of the robot

3 Actuator Design and Actuation Mechanism

The cooling module integrated SMA spring has a flexible tubing that is threaded through each of its spring coil. The SMA spring is resistively heated and the tubing acts as a coiled channel for the passage of water and air. The water cooling strategy mainly addresses two challenges: the low bandwidth of SMA and the usually bulky design of SMA when paired with a cooling mechanism.

The SMA spring (NiTiNOL springs from eBay.com) has a spring wire diameter of 0.75 mm and a mean coil diameter of 6.5 mm. Table 1 shows the tubings that we attempted to use as the cooling channel for the SMA spring. The small inner diameter and the high Shore A value of Tubing (1) make the threading process complicated. Though a thin wall of tubing (2) allows the addition of minimal stress on the SMA spring, it leads to easy tearing of the tubing. The thick wall of tubing (3) results in the development of excessive stress on the SMA spring and thus increases the transformation temperatures. Tubing (4) is attractive due to its softness but its inner diameter does not allow smooth water flow during SMA contraction. Tubing (5) therefore is our final choice because it has an inner diameter large enough for smooth water flow during the contraction and relaxation phase of SMA. It also has a wall thick enough to prevent easy tearing and at the same time thin enough to prevent the addition of excessive stress.

The threading process starts with the straightening of the two ends of the spring, which allow easy entry of the tubing. Upon having the tubing cover the entire spring length, electrical wires are attached to the spring ends and led to a circuit board for power connection. T-fittings are connected to the tubings at the straights ends of the spring and sealed with rubber plugs. Monofilament wires are connected at the spring ends and act as tendon wires that connect to the robot joint. Figure 3 shows a magnified view of a cooling module integrated SMA spring.

In the current work, water is used to cool the SMA spring during the cooling phase while compressed air at a gauge pressure of 25 Psi is used to remove water

Table 1 Tubing parameters

No.	Name	Inner diameter (in)	Outer diameter (in)	Wall thickness (in)	Softness (Shore A)
1	High purity white silicone rubber tubing (Non-reinforced)	0.040	0.085	0.023	55
2	Odor resistant white silicone rubber tubing	0.062	0.095	0.017	50
3	High temperature NSF-51 silicone rubber tubing	0.0625	0.125	0.03125	50
4	High temperature silicone rubber tubing	0.0625	0.125	0.03125	35
5	High purity white silicone tubing	0.078	0.125	0.024	50

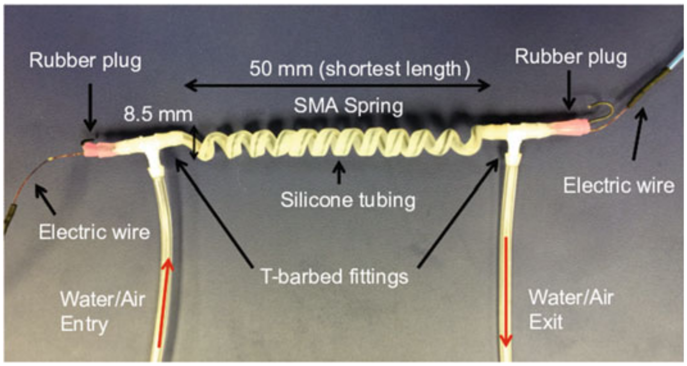


Fig. 3 SMA spring actuator with cooling module

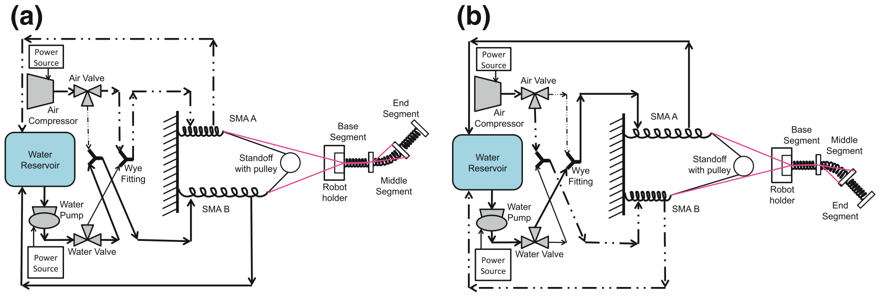


Fig. 4 **a** Schematic showing water flow (*bold solid line*) and air flow (*bold dotted line*) to bend middle segment upward; **b** schematic showing water flow (*bold solid line*) and air flow (*bold dotted line*) to bend middle segment downward

that remains in the tubing during the heating phase. Figure 4 shows the setup for one-DoF motion of the robot segment consisting of one antagonistic pair of SMA springs. When SMA A is heated to bend the middle segment upwards, water is allowed to flow through the cooling module in SMA B to cool the SMA spring. At the same time, the air valve is turned on for the initial 2 s, allowing the compressed air to be pumped into the cooling module in SMA A to force out water for a more efficient heating process.

4 Characterization of Actuator Performance for Maximum Actuation Bandwidth

Maximum actuation bandwidth is defined as the maximum frequency that an antagonistic SMA springs can be actuated when the corresponding robot segment moves between two target displacements for one complete cycle. The performance of SMA spring actuators in terms of maximum actuation bandwidth is dependent on several parameters in the current setup. The rate of change in temperature of the SMA spring is related to the current supplied and the flow rate of water. The pre-strain of SMA springs, gauge pressure of the air used for forcing out water and the motion amplitude over which the robot joint moves are also investigated as factors that affect the SMA performance. We performed characterization experiments by actuating the middle segment of the robot in the horizontal plane. The parameters at their control state are as follows: current = 4 A; water flow rate = 0.1 L/min; pre-strain = 50 mm; gauge pressure = 25 psi; motion amplitude = 10.5°.

4.1 Water Velocity

Heat transfer coefficient, h_w, can be defined as [18]:

$$h_w = \frac{k_w Nu}{D_h} \tag{1}$$

and the hydraulic diameter, D_h, can be expressed as $D_h = d_t - d_s$ [18]. d_t, d_s, Nu, and k_w are the silicone tubing inner diameter, SMA spring wire diameter, Nusselt number, and thermal conductivity of water respectively. During forced water convection, Nu can be expressed as a function of the Reynolds number, Re, which is given by:

$$Re = \frac{u_w D_h}{\nu_w}. \tag{2}$$

where u_w and ν_w are water velocity and kinematic viscosity, respectively. Water velocity is varied to change the Reynolds number and therefore the heat transfer coefficient. In the characterization experiments, we attempted five water flow rates: 0.05, 0.1, 0.2, 0.3, and 0.4 L/min.

4.2 Current

Current is provided to heat the SMA spring through a motor driver that is operated in its current controller mode and is powered by a power source that provides 24 V, which is large enough to ensure the supply of the desired current. An upper limit of the current is set at 4.2 A to ensure that the SMAs are not overheated. We attempted four different currents, namely 3.5, 3.8, 4.0, and 4.2 A.

4.3 Gauge Pressure

The compressed air is pumped into the cooling channel at different gauge pressures, including 5, 15, 20, and 25 psi. The different air speeds force the static water in the cooling module to flow out at different speeds. The hypothesis is that water can be removed more quickly when a higher gauge pressure is used, thus allowing efficient heating to take effect earlier during each heating phase. This can potentially lead to the SMA springs having a higher actuation bandwidth.

4.4 Pre-strain

Pre-strain of antagonistic SMA springs can affect the actuation bandwidth since the internal stress of SMA affects its transformation temperatures [19]. The range of motion of each SMA is maintained between its austenite displacement and its maximum recoverable displacement, which has been experimentally determined to be 80 mm [10]. The control pre-displacement of each SMA was chosen to be 50 mm to ensure a large range of motion of each antagonistic SMA spring in either direction. We attempted three different pre-strains of 45, 50 and 55 mm and investigated the effect of different pre-strains on the actuation bandwidth of SMA springs on a continuum robot.

4.5 Motion Amplitude

Motion amplitude determines the distance that a robot segment has to go through. A larger amplitude requires the SMA to be heated to a higher temperature. Therefore, it is important to associate the actuation bandwidth of SMAs with motion amplitude of the robot. Four different angular displacements of the robot segments were attempted, namely ±5.5°, ±10.5°, ±15.5°, and ±20.5°.

5 Vision-Based Experimental Setup

The experimental setup for MINIR-II consists of a robotic platform, an air compressor, a water reservoir, electrical circuit, a computer with an Analog/Digital board, a stereo camera, and eight water and air valves. The robotic platform, shown in Fig. 5, is made out of acrylic plates and consists of a robot holder for holding the MINIR-II, a back wall to which the springs are fixed on one end, an intermediate wall to guide the tendons, and a top plate that prevents tubings from interfering with neighboring SMA springs. Eight SMA springs are arranged in two rows and they

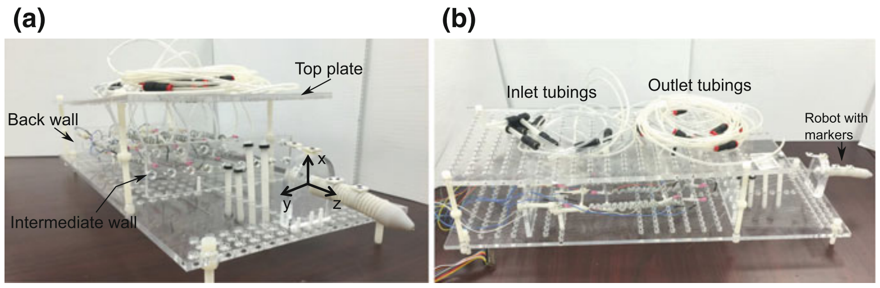


Fig. 5 **a** Robotic platform with the spring-based robot, eight cooling module integrated SMA springs, and tubings. **b** Side view of the robotic platform made out of laser-cut acrylic

are connected to the robot joints via tendons. Each antagonistic pair of SMA springs are also connected directly to each other via a tendon that is routed around a pulley sitting on a standoff. This is done to ensure that the non-heated SMA spring always applies zero tension on the robot segment and that the flexible spring-based robot is not compressed due to unexpected tension from the non-heated SMA spring. The 4-DoF robot motion tracking is controlled via a vision-based setup, in which four markers were attached on the robot, as shown in Fig. 5. A vector is formed between adjacent markers and angular displacement of each robot segment is calculated from the angle of the corresponding vector relative to its previous vector.

In the characterization experiments, we applied step inputs of $\pm 10.5^\circ$ to the middle segment in the y–z plane. All the parameters were set in the control state except the parameter that was being varied. Experiments were also done to test the performance of each of the base and middle segment. The base segment was actuated in the x–z plane while the middle segment was actuated in the y–z plane. Lastly, an experiment was done on the base and middle segment of the robot to test the coordinated motion between two segments, independent joint controllability, and repeatability of the robot motion. The middle segment was commanded to move to a step input of $+10.5^\circ$. This is followed by the base segment moving to a step input of -10.5°. Then the base segment and middle segment were commanded to return to 0° one after another.

6 Results and Discussion

6.1 *Characterization for Optimal Performance*

The results from the characterization experiments show how the middle segment move between $\pm 10.5^\circ$. In the following discussion, the rise time is defined as the time required to move from -10.5° to $+10.5^\circ$ while the fall time is the time required

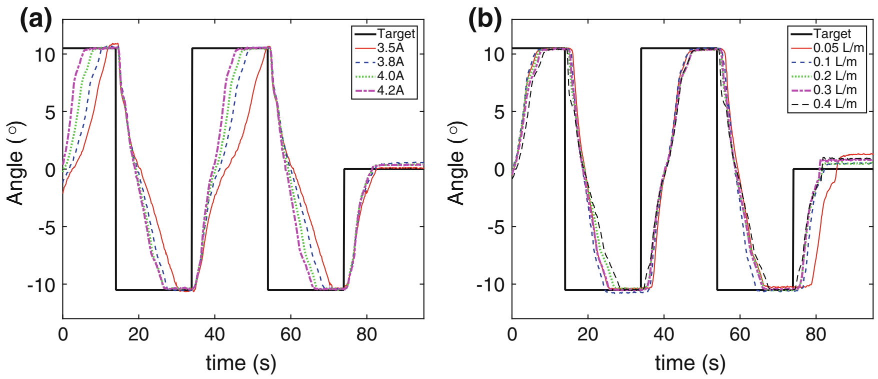


Fig. 6 Effect of **a** current and **b** water flow rate on the change of angular displacement versus time

to move from +10.5° to −10.5°. An actuation period is the sum of the rise time and the fall time, and actuation bandwidth is the reciprocal of the actuation period.

Figure 6a shows that the rise times are 18.85, 16.03, 14.73, and 13.3 s and fall times are 17.05, 14.6, 13.14, and 12.64 s for currents of 3.5, 3.8, 4, and 4.2 A, respectively. The actuation period decreases with an increase in current. The major difference in actuation period between the different currents occurs in the second half of each rise and fall curve. This implies that current mainly increases the heating rate and has little effect on the cooling rate.

Figure 6b shows that the rise time and fall time are between 13.1 and 15.1 s and between 10.41 and 14.68 s for water flow rates of 0.05, 0.1, 0.2, 0.3, and 0.4 L/min. The highest flow rate of 0.4 L/min led to the highest rate of change in angular displacement in the beginning of each rise and fall curve. However, the low temperature it cooled the SMA to demanded additional heating time, leading to eventually the longest actuation period. The optimal flow rate, based on the shortest actuation period, was therefore 0.1 L/min. It cooled the SMA to approximately its martensite finish temperature, where SMA is at its lowest stiffness, and subsequently allowed the minimum amount of heating time to contract the SMA to its target displacement.

The effect of pre-strain is not distinct in the plot of Fig. 7a. However, through careful analysis, the maximum pre-strain corresponded with the highest average actuation period of 24.75 s while the minimum pre-strain led to the shortest actuation period of 23.65 s. This agrees with previous literature by Tanaka [19] and Brinson [20], who suggested that higher pre-strain causes a highest shear stress in the SMA spring and therefore increases the transformation temperatures and the heating time.

As seen in Fig. 7b, gauge pressure of 5, 15, 20, and 25 psi led to actuation period of 30.01, 26.22, 23.51, and 22.89 s respectively. Different gauge pressure contributed to different rate of increase in the SMA temperature during the heating phases, which are represented by the second half of each rise and fall curve. The cooling rate in the first half of each rise and fall curve is not affected by varying the gauge pressure. Figure 8 shows that the average actuation bandwidth decreases with the motion amplitude.

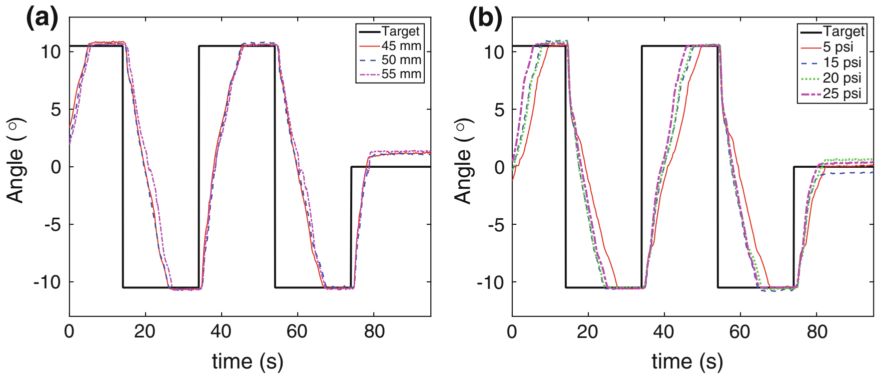
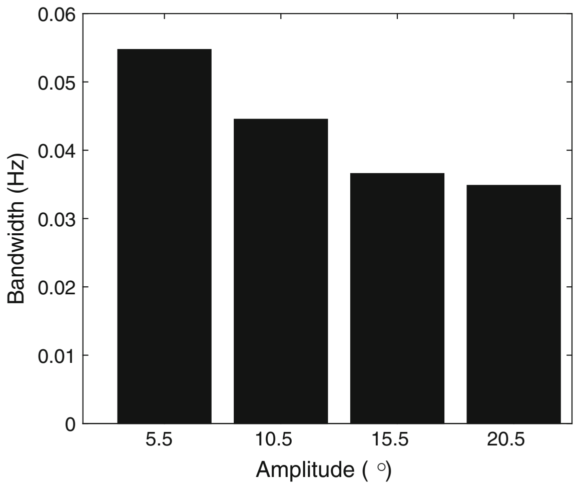


Fig. 7 Effect of **a** pre-strain and **b** gauge pressure on the change of angular displacement versus time

Fig. 8 Bandwidth as a function of motion amplitude



SMA springs are required to be heated to a higher transformation temperatures to achieve larger motion amplitudes. The inherent non-linear property of SMA is also confirmed by the experimental results. The small initial strain change with temperature led to relatively big difference in the motion bandwidth between amplitude of 5.5° and that of 10.5°. The larger strain change with temperature at higher SMA temperature caused the difference in actuation bandwidth between motion amplitude of 15.5° and 20.5° to be small.

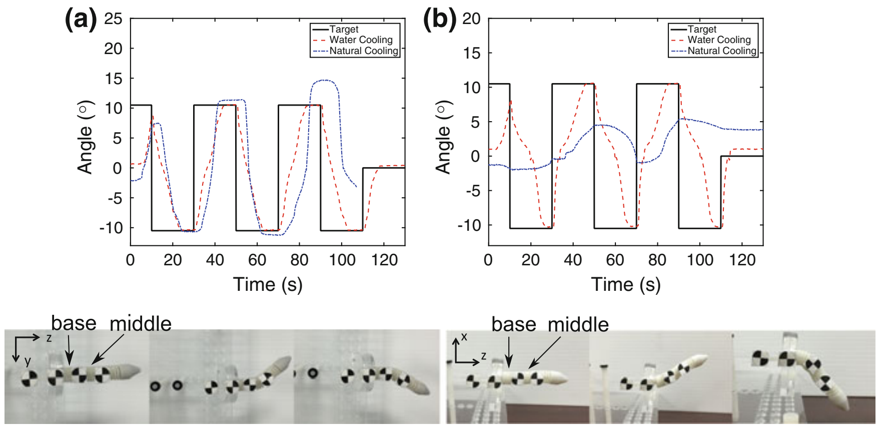


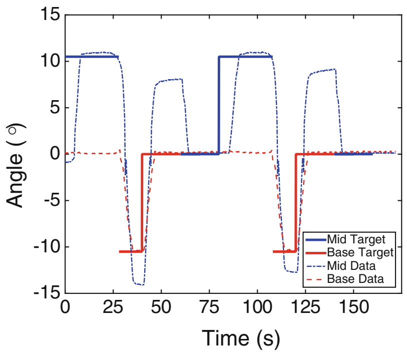
Fig. 9 **a** Motion tracking of middle segment and pictures showing it in home configuration, +10.5° and −10.5°. **b** Motion tracking of base segment and pictures showing it in home configuration, +10.5° and −10.5°

6.2 Robot Motion

The actuation performance of the robot segment under force water cooling was compared with that under natural air cooling. Figure 9a shows that the actuation period was 26.75 s for the middle segment when the antagonistic SMA springs were cooled under forced water convection. When they were naturally cooled, overshoot happened in almost every motion cycle. The tendon then broke at 107 s because under natural cooling, residual heat and temperature built up in the SMAs and induced excessive stress in the tendons.

The base segment was then actuated under forced water cooling and natural air cooling. A smaller current was applied to prevent overshoot and stress from building up quickly in the SMAs. However, it did not allow the SMAs to be actuated fast enough, leading to poor tracking of the robot segment with respect to the reference step inputs. Under forced water cooling, the base segment achieved an actuation period of 32.1 s. There is a discrepancy between actuation period in the base segment and that of the middle segment because the cooling modules were hand manufactured for each SMA and therefore could lead to different performance outcome.

Lastly, the base and middle segments were alternately actuated and one target step input was provided to one segment at any instant. The robot was able to achieve independent joint control since the base segment stayed unmoved at 0° when the middle segment moved towards +10.5° reference angle. The base segment was then actuated to achieve −10.5° before being commanded to return to 0°. The middle segment was eventually returned to its home configuration of 0°. The entire process was repeated to ensure the repeatability of the robotic system (Fig. 10).



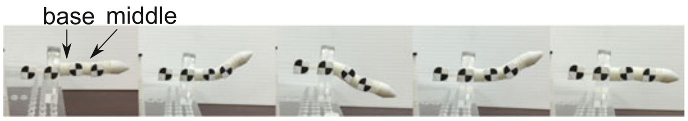


Fig. 10 Motion tracking result of both base and middle segments while actuating them alternatively; Picture showing one complete cycle of motion

7 Conclusion

A robotic platform has been developed with the eventual goal of being used for an intraoperative MRI-guided neurosurgical procedure. The flexible spring-based three-segmented MINIR-II prototype, manufactured from a 3-D printer, has four active DoFs with the base and middle segment actuated by four pairs of SMA spring and the end segment unactuated. Due to the low cooling rate under natural air cooling, each SMA springs was covered in a silicone tubing, reasonably selected based on the diameters and hardness of the tubing, to form a cooling module integrated SMA actuator. Water is flowed through to cool the SMA during the cooling phases while compressed air is passed through the cooling module to force out water during the heating phases to improve heating efficiency. The antagonistic pair of SMA springs connected to the middle segment of the robot was characterized using five parameters, namely the current, water flow rate, pre-strain, gauge pressure, and motion amplitude. We determined that a higher current increases the heating rate and thus the actuation bandwidth of the SMAs. A higher water flow rate provides a greater cooling rate but does not necessarily improves the actuation bandwidth. In our case, a flow rate of 0.1 L/min was optimal at cooling the SMA close to its martensite finish temperature, subsequently allowing a short heating time for the SMA to reach its target displacement. The experimental result shows that varying pre-strain would not significantly contribute to improvement of the actuation bandwidth but a higher pre-strain still increases, despite slightly, the transformation temperatures and therefore the rise and fall times. Higher gauge pressure increases the force of the air that eliminates water from the cooling module, leading to quicker and more efficient heating phases. The motion of the base and middle robot segments were evaluated and compared under the case of natural air cooling and that of forced water cooling. Fast actuation of SMA springs is not sustainable under natural air cooling since stress easily builds up in the SMAs due to the residual heat that is not completely removed during the cooling phases. In our future work, we will investigate the change in force with time to make sure the robot can apply sufficient tip force when actuated at higher bandwidth. We will theoretically model the optimal flow rate and compare it with the experimental data. We will also compare the effectiveness of forced water cooling and forced air cooling under the same experimental condition, and attempt a more compact design of the robotic platform for the full 6-DoF robot.

Acknowledgements Research reported in this publication was supported by the National Institute of Biomedical Imaging And Bioengineering of the National Institutes of Health under Award Number R01EB015870. The content is solely the responsibility of the authors and does not necessarily represent the official views of the National Institutes of Health.

References

1. Cancer survival rate statistics by type of cancer. Disabled World (2010)
2. Bindal, A.K., Bindal, R.K., Hess, K.R., Shiu, A., Hassenbusch, S.J., Shi, W.M., Sawaya, R.: Surgery versus radiosurgery in the treatment of brain metastasis. J. Neurosurg. **84**(5), 748–754 (1996). PMID: 8622147
3. Bajo, A., Simaan, N.: Kinematics-based detection and localization of contacts along multisegment continuum robots. IEEE Trans. Robot. **28**(2), 291–302 (2012)
4. Webster, R.J., Romano, J.M., Cowan, N.J.: Mechanics of precurved-tube continuum robots. IEEE Trans. Robot. **25**(1), 67–78 (2009)
5. Choi, D.G., Yi, B.J., Kim, W.K.: Design of a spring backbone micro endoscope. In: Proceedings of the IEEE/RSJ Intelligent Robots and Systems (IROS 2007), pp. 1815–1821. IEEE (2007)
6. Jani, J.M., Leary, M., Subic, A., Gibson, M.A.: A review of shape memory alloy research, applications and opportunities. Mat. Des. **56**, 1078–1113 (2014)
7. Ho, M., McMillan, A.B., Simard, J.M., Gullapalli, R., Desai, J.P.: Toward a meso-scale SMA-actuated MRI-compatible neurosurgical robot. IEEE Trans. Robot. **28**(1), 213–222 (2012)
8. Ho, M., Desai, J.P.: Modeling, characterization and control of antagonistic SMA springs for use in a neurosurgical robot. In: Proceedings of the IEEE International Conference on Robotics and Automation (ICRA 2013), pp. 2503–2508, May 2013
9. Tadesse, Y., Thayer, N., Priya, S.: Tailoring the response time of shape memory alloy wires through active cooling and pre-stress. J. Intell. Mat. Syst. Struct. **21**(1), 19–40 (2010)
10. Cheng, S.S., Desai, J.P.: Towards high frequency actuation of SMA spring for the neurosurgical robot-MINIR-II. In: Proceedings of the IEEE International Conference on Robotics and Automation (ICRA 2015), pp. 2580–2585. IEEE, May 2015
11. Fukami, N.: Endoscopic Submucosal Dissection: Principles and Practice. Springer, Heidelberg (2015)
12. Louw, D.F., Fielding, T., McBeth, P.B., Gregoris, D., Newhook, P., Sutherland, G.R.: Surgical robotics: a review and neurosurgical prototype development. Neurosurgery, 54(3) (2004)
13. Comparetti, M.D., De Momi, E., Vaccarella, A., Riechmann, M., Ferrigno, G.: Optically tracked multi-robot system for keyhole neurosurgery. In: Proceedings of the IEEE International Conference on Robotics and Automation (ICRA 2011), pp. 661–666, May 2011
14. Lewis, N., York, A., Seelecke, S.: Experimental characterization of self-sensing SMA actuators under controlled convective cooling. Smart Mat.Struct. **22**(9), 094012 (2013)

15. Loh, C.S., Yokoi, H., Arai, T.: Natural heat-sinking control method for high-speed actuation of the SMA. Int. J. Adv. Robot. Syst. 3(4) (2006)
16. Andrew Russell, R., Gorbet, R.B.: Improving the response of SMA actuators. In: Proceedings of the IEEE International Conference on Robotics and Automation (ICRA 1995), vol. 3, pp. 2299–2304, May 1995
17. Ertel, J., Mascaro, S.: Dynamic thermomechanical modeling of a wet shape memory alloy actuator. J. Dyn. Syst. Meas. Control **132**, 45–57 (2010)
18. Incropera, F.P.: Fundamentals of Heat and Mass Transfer. John Wiley & Sons, New York (2006)
19. Tanaka, K.: A thermomechanical sketch of shape memory effect: one-dimensional tensile behavior. Res. Mech. **18**(3), 251–263 (1986)
20. Brinson, L.C.: One-dimensional constitutive behavior of shape memory alloys: thermomechanical derivation with non-constant material functions and redefined martensite internal variable. J. Intell. Mat. Syst. Struct. **4**(2), 229–242 (1993)

Variable Stiffness Pneumatic Structures for Wearable Supernumerary Robotic Devices

Frank L. Hammond III, Faye Wu and H. Harry Asada

1 Introduction

Millions of people in the U.S. suffer from impaired hand function due to a variety of medical conditions including stroke-induced neurological injury, blunt physical trauma, and various diseases that require surgical modification of the hand anatomy [20]. The disruption or loss of hand function caused by these conditions can, depending upon the severity, prevent a patient from performing everyday grasping and manipulation tasks and can significantly reduce a patient's quality of life. Conventional physical rehabilitation is very often an effective means of restoring function to the hands of patients suffering from moderate neurological injury (Krebs 1999; [30]), but for patients who suffer from severe neurological injury or whose hands have been disfigured by disease or physical trauma, wearable robotic motion assist devices and robotic prostheses are sometimes the only means of enabling the patient to accomplish activities of daily living (ADL) [11, 14, 17].

Several research groups are developing wearable robotic grasp assist devices to address the needs of patients suffering impaired hand function, and each solution has its own merits and drawbacks. Many wearable grasp assist solutions take the form of rigid exoskeletons which provide assistive forces across various patient joints [2, 16]. Exoskeletal devices, which are typically cable or gear driven, are known to provide precise, controllable, high-force motions required to grasp daily living objects, and have shown efficacy in improving patients' hand functionality in rehabilitation studies

F.L. Hammond III (✉) · F. Wu · H.H. Asada
Department of Mechanical Engineering, MIT 77 Massachusetts Avenue, Cambridge, MA 02139, USA
e-mail: fhammond@gatech.edu

F. Wu
e-mail: yfwu@mit.edu

H.H. Asada
e-mail: asada@mit.edu

[5, 12, 18, 33]. However, such devices do not accommodate variations in patient skeletal structure or joint misalignments and can produce compression forces on the soft tissue and joints during long-term use [26] unless special design considerations are made [6]. In addition, exoskeletal devices can limit patient motion range and be difficult to don due to device size, and making them unfit for clinical applications [12, 33].

In contrast to rigid exoskeleton motion assist devices, soft wearable robotic devices are innately compliant, backdrivable alternatives which easily conform to the body of the wearer and safely provide the motive forces required to actuate human joints. Recent advances in soft robot fabrication techniques have enabled the creation of fluidic, muscle like actuators [13, 15, 21, 24, 29] that can be embedded in soft wearable devices and can provide the assistive forces using low mass, high throughput pressurized air sources. Studies have demonstrated that pneumatically-driven soft bending actuators could be used to create a low-cost, lightweight wearable glove capable of assisting basic hand closing motions [7, 23], and that soft actuators can also be programmed using mechanical constraints and multiple chambers to produce specific actuation profiles [10]. However, due to the same intrinsic flexibility that lends to their safety, pneumatic wearable devices are less precise and harder to control and coordinate than rigid exoskeletons. Also, in cases where these devices are constructed from continuous-body inflatable structures, large amounts of gas are required to produce motion and maintain device shape, making lightweight, portable power sources less feasible. The monolithic structure also makes adaptation to anatomical variations challenging.

Supernumerary robotic (SR) devices are a class of wearable device which adds extra limbs to the user to enhance manipulation capabilities. These devices enhance manipulation capability without relying on the user's skeletal structure for support, making anatomical variation and motion restriction a lesser issue. Recent work on supernumerary fingers for human augmentation [31] demonstrated the efficacy of SR devices to compensate for lost motion in the human hand by providing grasp affordances and greater grasping forces. Despite the functional benefits of these SR fingers, their use of powerful electric motors and rigid components present user safety issues that limit clinical suitability.

In this paper, a pneumatic supernumerary robotic (SR) finger device is proposed as a means of improving the portability, controllability, and configurability of grasp assist devices (Fig. 1). The proposed grasp assist device is composed of (1) modular, inflatable structures which reduce pneumatic power source requirements, and (2) tunable bending actuators which make grasp motion programming and device control less challenging. The modular nature of the pneumatic SR fingers and the ability to modify the joint motion rates for programmable synergies can make wearable grasp assist devices more economical (manufacturing cost and energy efficiency), more adaptive (reconfigurable for different patients and tasks) and more feasible as a solution for long-term treatment of impaired hand function.

The major contribution of this paper is the design, analysis, and experimental characterization of the pneumatic SR finger's two primary components. First, an inflatable robotic finger phalanx - based on strain limited pneumatic bladders [10] - is designed

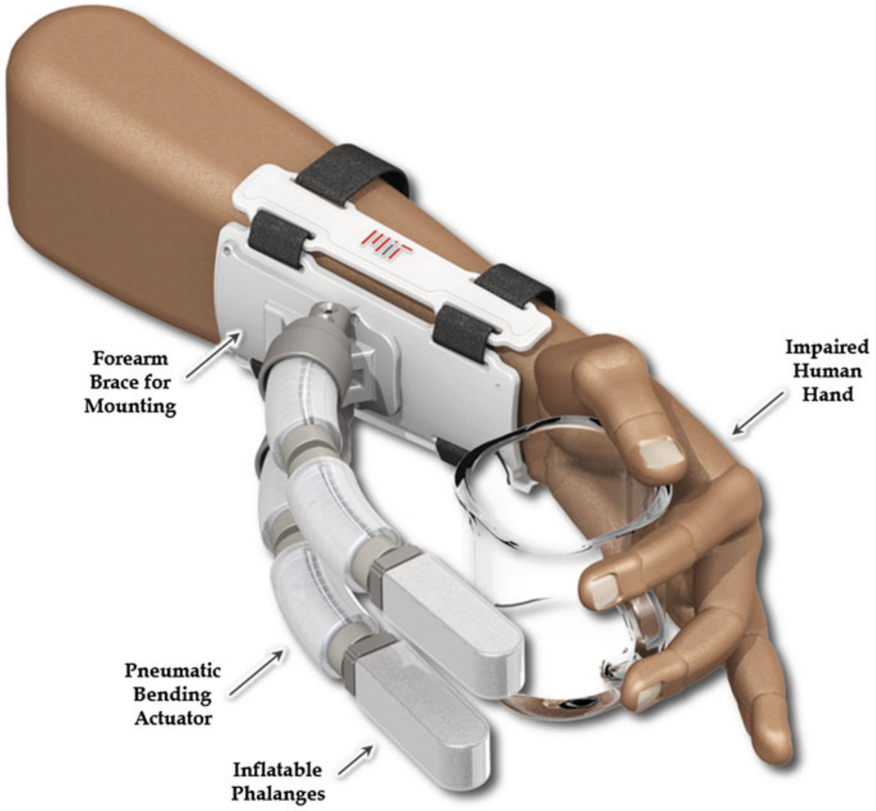


Fig. 1 Pneumatic supernumerary robotic fingers being used to provide grasp assistance to individuals suffering from stroke-induced hand impairment

to inflate to its functional form from a lower pressure smaller-volume storage state and accommodate the grasp forces associated with ADL. Second, a dual-chamber, bidirectional pneumatic bending actuator is designed to enable programmable mechanical stiffness and tunable motion rates for the SR finger joints. Fabrication methods and experimental characterizations of mechanical behavior (deformation, motion) for each of the two components are presented. Finally, the bidirectional actuators and inflatable phalanges are assembled in a wearable grasp assist device and tested to demonstrate the ability to strategically program joint motion patterns for grasping objects of daily living.

2 Programmable Pneumatic Grasp Assist Device: Performance Requirements and Relevant Functional Features

2.1 Clinical Motivation

Research has shown that early implementation of rehabilitation practices is critical to the recovery of patients suffering from impaired hand function. Central to the adoption and efficacy of these practices is the availability of affordable, portable, easy-to-use rehabilitation devices that not only allow patients and clinicians greater treatment flexibility, but also target the grasping and manipulation capabilities most essential to patient recovery [11, 14]. With proper assistive devices, patients can more quickly begin to recover motor function and increase their mobility, granting them greater independence and improved quality of life.

2.1.1 Simple Grasping Capabilities Are Essential to Mobility

The standard measure of patient recovery from hand impairment is their ability to participate activities of daily living (ADLs). Many of the objects used in ADLs, such as cups, bowls, and door knobs, require stable pinch or power grasps - not the dexterous manipulation capabilities associated with fine motor skills in healthy subjects. Such grasps can be supported by devices which have limited degrees of freedom, low precision, and a kinematically simple structure [4, 9]. To this extent, complex assistive devices with precision motors, large power sources, and complicated control strategies are not essential to treatment. Simple, low-precision, elastomeric grasp-assist devices are a sufficient solution.

2.1.2 Accommodation of Anatomical and Pathological Variation

Given the large variation in degrees of patient impairment and the anatomical configuration of their hands, a proper grasp assist device must be highly reconfigurable – both kinetically and kinematically - for effective patient-specific solutions. SR devices, unlike exoskeletal, assistive devices [23], rely on coordinated motion between “extra” robotic limbs and existing biological limbs [19, 25, 32] rather than on providing mechanical support to existing limbs. In theory, an SR grasp assist device could aid in the treatment of patients either with or without intact anatomy and regardless of their specific motor deficiency, making this type of device suitable for a wider range of patients and disabilities than exoskeletons.

2.2 Performance Requirements

For a wearable robotic grasp assist device to be fit for clinical applications in rehabilitation and the treatment of severe, long-term functional hand impairment, the device must meet several human factors and performance criteria. These criteria, taken from literature and from discussions with rehabilitation experts, include:

- **Energy efficiency**: Not requiring expensive, unsafe, cumbersome energy sources (large batteries, pumps).
- **Low-encumbrance**: Lightweight (<500 g) and ergonomic for long-term use on patients with chronic disabilities
- **Durability**: Must accommodate a large number of usage cycles in daily living environments while resisting damage from heat, puncture, etc.
- **Functional versatility**: Accommodates a wide variety of objects and tasks associated with ADLs [11].
- **Configurability**: Easy to program for patient-specific needs, including range and types of objects and anatomical variations (e.g. no. of functional digits).
- **Mechanical power**: Capable of generating the contact forces ($\sim$5N) required for successfully grasping objects of daily living.
- **Error tolerance**: Robust to positioning and sensing errors, and having mechanical design features suited to grasp stability including: high friction, natural distributed compliance, underactuation [9].

A review of literature on soft robotics and wearable devices suggests that durable, powerful, low-encumbrance grasp assistance devices are attainable using current soft fabrication and actuation methods [8, 23]. The functional versatility, configurability, controllability, and energy efficiency of wearable grasp assist devices, however, are performance criteria which are predicated more on design strategy than on technical, manufacturing capabilities. The proposed SR device is focused on meeting these criteria, and this work provides foundation for predicting the behavior of inflatable structures used in the device.

2.3 Grasp Assist Device Design

2.3.1 Design Rationale

The proposed pneumatic SR grasp assist device is comprised of two SR fingers, each with two proximal bending actuators connected in series and one distal phalanx serving as the primary object-robot interface (Fig. 2). Actuators and phalanges are 73 and 70 mm in length respectively. A single actuator in series with a phalanx is 143 mm in length, roughly the size of a small female hand. Two actuators in series with a phalanx are 216 mm long, about the size of a large male hand.

Each actuator has a primary and an antagonistic chamber. Individuation of chamber pressure control allows for eight separate pressure sources, but to simplify the

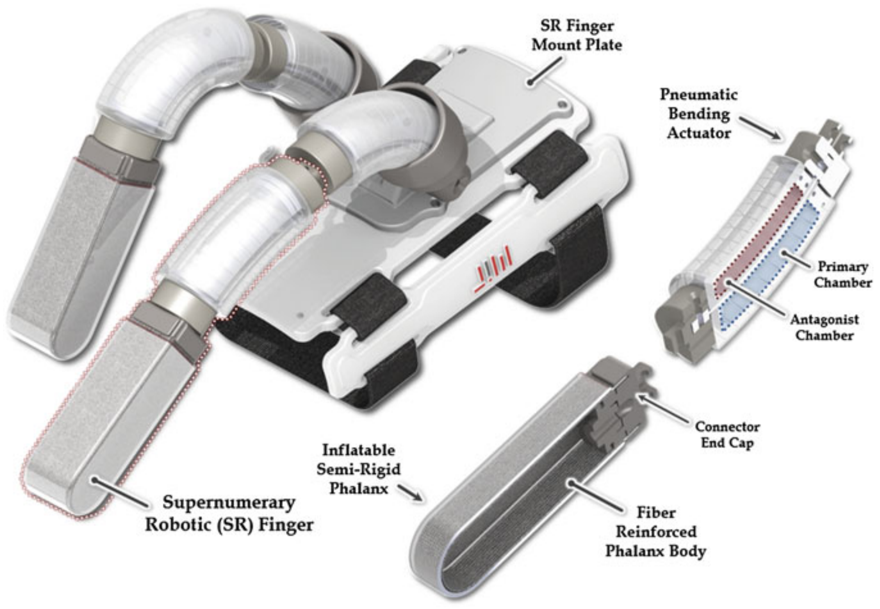

SR Finger Mount Plate
Pneumatic Bending Actuator
Primary Chamber
Antagonist Chamber
Connector End Cap
Inflatable Semi-Rigid Phalanx
Fiber Reinforced Phalanx Body
Supernumerary Robotic (SR) Finger

Fig. 2 The proposed pneumatic SR grasp assist device and its modular components: bidirectional pneumatic bending actuators and inflatable, semi-rigid phalanges

operation of the device and to leverage the benefits of underactuation (i.e. passive compliance for robust grasp acquisition), the primary actuator chambers share the same pressure source and the antagonist chambers are pressurized individually with a second pressure source. To "program" motion synergies into the motion of the SR grasp assist device, the antagonist chambers are pressurized to pre-determined levels that affect how quickly, and with what mechanical force, the actuators bend with respect to the shared primary chamber source. This device, in essence, becomes a fluidic version of the cable-pulley mechanisms used for passive underactuated grasp acquisition in previous work [9].

The proposed design has several functional benefits, including the following:

- **Energy efficiency through targeted inflation/deflation**: Inflatable phalanges, which comprise a large portion of finger inflation volume, inflate once during deployment and are then held at a constant operational pressure. Only actuators require cyclic inflation, reducing total power and time requirements.
- **Modularity and reconfigurability**: The SR finger components can be assembled in various configurations to create fingers of different sizes and actuation capabilities, allowing for more flexible, patient-specific customization of the devices. Components can also be manufactured, tested, and repaired individually, making assembly and modification of wearable devices simpler.
- **Underactuation**: Pneumatic SR finger compliance and the ability to actuate several chambers in parallel allows an SR grasp assist device to operate as an

underactuated grasping device – proven in literature as adept at grasping a variety of objects without low-level control [22]. A single compressed gas source can inflate all of the primary chambers in the SR grasp device, while strategically inflated antagonist chambers modulate actuator motions at different rates. This concept is similar to work in [9, 22] where compliant flexures of different bending stiffness were used to modulate motion patterns of a cable-driven underactuated robotic hand. For pneumatic SR fingers, however, the mechanical stiffness and motion rates of the joints can be controlled in real-time by changing antagonist actuator pressure, rather than by physically swapping joints to change stiffness. The size and shape of graspable objects varies with phalanx size, actuator motion range, kinematic topology, and SR device position on the wearer.

3 Modeling the Elastic Deformation Mechanics

The pneumatic bending actuator and inflatable phalanx are comprised of both strain limiting elements and elastomeric substrates. The highly nonlinear mechanics of elastomer deformation make analytical solutions very challenging, but simplified mechanistic models and empirical data can be used to provide a rudimentary basis for design and control of the inflatable SR finger structures.

3.1 Inflatable Semi-rigid Phalanx

The inflatable semi-rigid phalanx model is simplified as a thin-walled pressure vessel with zero tensile strain (due to strain limiters). A section through the midline of the vessel creates a planar geometry with thin beams and a rounded cap (Fig. 3). To consider the inflatable phalanx's stiffness against off-axis normal forces applied at the tip (causing tip deflection and eventually buckling), we model the problem as a pair of elastic columns (surfaces of a cantilever), one in compression and one in tension. We assume that phalanx buckling can occur only after the bending-induced compressive stress in a column is greater than the internal pressure-induced tensile stress. Total stress σ_{tot} in the column is the sum of cantilever stress σ_F and longitudinal pressure vessel stress, σ_l, and the minimum pressure required to prevent the onset of buckling is given by Eq. 1.

The force F_T on the phalanx tip that generates the critical Euler buckling force F_B for a given geometry, internal pressure, and normal force is approximated by Eq. 2, where end condition factor $n = 4$ (Fig. 3). Area A is the cross section of three of four phalanx walls in compression to approximate all material in compression and resisting buckling (top wall is always tensioned). Setting F_B/A equal to total column stress, we find F_T in terms of combined stresses. We assume for this model that phalanx internal pressure$P$ remains constant during deformation.

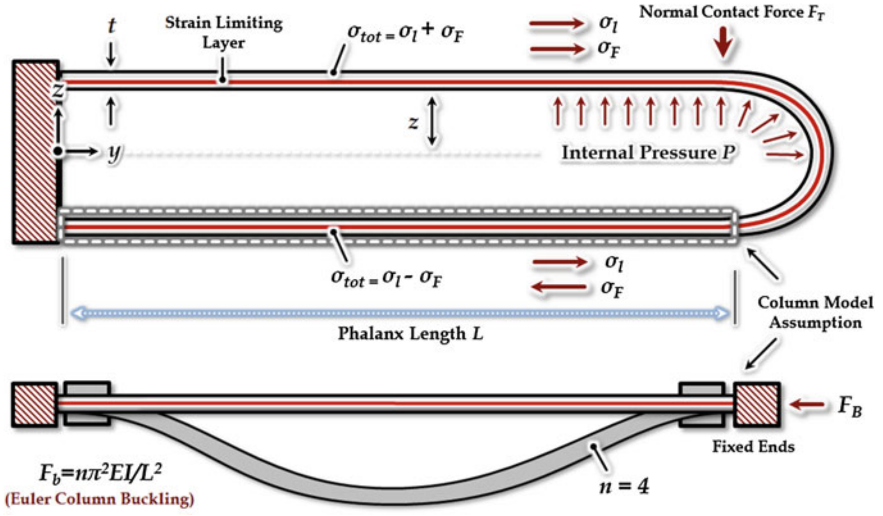


Fig. 3 A diagram of the phalanx internal pressure, normal loading forces and the approximation of phalanx buckling mechanics (*bottom*)

$$\sigma_F = \frac{Mz}{I} \quad , \quad \sigma_l = \frac{Pz}{2t}; \quad \sigma_{tot} = \frac{Pz}{2t} + \frac{Mz}{I} \quad \therefore \quad P_B = \frac{-2Mt}{I} = \frac{-2F_T Lt}{I} \tag{1}$$

$$F_B = \frac{n\pi^2 EI}{L^2} \quad ; \quad \frac{F_B}{A} = \frac{n\pi^2 EI}{L^2 A} = \sigma_{tot} \quad ; \quad \frac{n\pi^2 EI}{L^2 A} = \frac{Pz}{2t} + \frac{F_T Lz}{I}$$

$$F_T = \frac{I}{Lz}(\frac{n\pi^2 EI}{L^2 A} - \frac{Pz}{2t}). \tag{2}$$

3.2 Pneumatic Bending Actuator

Analytical solutions have been derived to predict the deformation mechanics of fiber reinforced pneumatic bending actuators [23]. Most mechanistic models describe a single chamber, hemispherical bending actuator and explain bending in one direction. These models are not easily extensible to the complex geometry and fiber reinforcements of the bidirectional bending presented here. Therefore, the characterization of bidirectional actuator deformation mechanics is done experimentally for the purposes of this preliminary work.

4 Soft Inflatable Component Fabrication Methods

Fabrication of the inflatable SR finger components involves a multi-step mold-based process in which elastic and inelastic constituents are mechanically integrated. This process is based in part on previous work in soft robot fabrication [3, 10, 23, 28].

4.1 *Inflatable Semi-rigid Phalanx Fabrication*

The inflatable semi-rigid phalanges are manufactured using a four-step process. An exploded view of the phalanx components is shown in Fig. 4.

Step 1: A strain limiting outer layer is created by embedding woven fiberglass fabric (US Composites Inc.) with a two part silicone (DragonSkin 30, Smooth-On Inc.). The fabric is cut into the desired pattern with a laser engraver (Universal Laser Systems Inc), wrapped around a phalanx-shaped mold plug and secured using in a silicone adhesive (SilPoxy, Smooth-On Inc.).

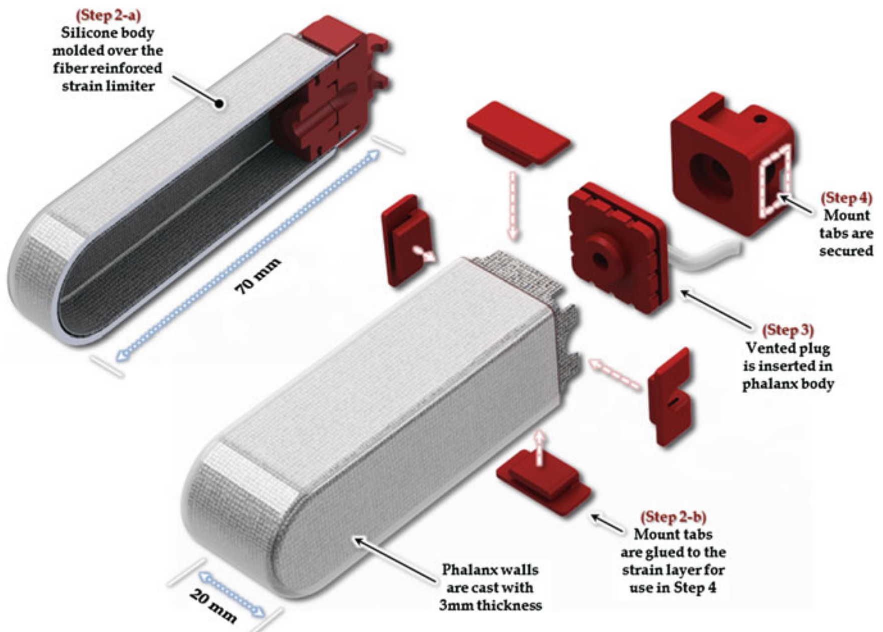

(Step 2-a)
Silicone body molded over the fiber reinforced strain limiter
70 mm
(Step 4)
Mount tabs are secured
(Step 3)
Vented plug is inserted in phalanx body
(Step 2-b)
Mount tabs are glued to the strain layer for use in Step 4
Phalanx walls are cast with 3mm thickness
20 mm

Fig. 4 A sectioned view (*top left*) and exploded view (*bottom right*) of the components of the inflatable semi-rigid phalanx, with 3D printed parts colored *red*

Step 2: The mold plug is inserted into the phalanx mold and silicone (DragonSkin 30) is injected to form the body of the phalanx. After curing, mount tabs are glued to the strain limiting layer for connection to the end caps.

Step 3: Flexible PVC tubing is inserted into a 3D-printed vented phalanx plug and secured using high-strength epoxy. The plug-tube assembly is then inserted into the open end of the phalanx body after plug edges and the inside surface of the phalanx body have been wetted with silicone.

Step 4: The strain limiter mount tabs are secured to the connector end cap using high-strength epoxy. A small amount of silicone is injected into the seams around the tabs to prevent fluid penetration.

The completed phalanx is connected to a gas source using barbed tubing adapters, and to actuators or other components using the end cap mount block.

4.2 Bidirectional Bending Actuator Fabrication

Bidirectional bending actuators are manufactured using a seven-step mold-based process. An exploded view of the actuator components is shown in Fig. 5.

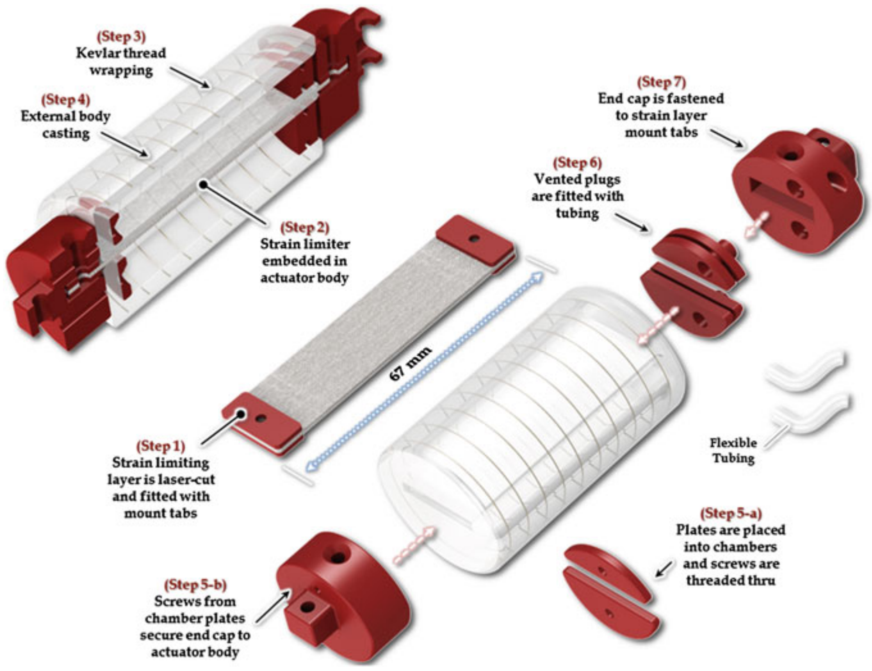

(Step 3) Kevlar thread wrapping
(Step 4) External body casting
(Step 7) End cap is fastened to strain layer mount tabs
(Step 6) Vented plugs are fitted with tubing
(Step 2) Strain limiter embedded in actuator body
67 mm
Flexible Tubing
(Step 1) Strain limiting layer is laser-cut and fitted with mount tabs
(Step 5-a) Plates are placed into chambers and screws are threaded thru
(Step 5-b) Screws from chamber plates secure end cap to actuator body

Fig. 5 Sectioned view (*top left*) and exploded view (*bottom right*) of bidirectional bending actuator components, with 3D printed parts colored *red*

Step 1: The strain limiting layer is created by the silicone-fabric embedding method used in the phalanx fabrication process. After being laser-cut to the desired shape, 3D-printed tabs are affixed to the ends of the strain limiting strip to prevent fabric tearing when loads are applied.

Step 2: The strain limiting layer is inserted into an actuator mold plug, which is then placed in the internal actuator body mold. Silicone (DragonSkin 10, Smooth-On Inc.) is injected into the mold and then degassed before curing.

Step 3: The actuator is removed from the internal body mold but left on the mold plug. Kevlar thread is wrapped around the actuator in a helical pattern, using notches in the actuator as guides, and is secured to the body using SilPoxy.

Step 4: The Kevlar wrapped actuator body is placed into a slightly larger (2 mm wider) external body mold and silicone is injected into the mold, covering the thread and locking it in place.

Step 5: After removal from the mold, the closed end of the actuator is fitted with an end cap by inserting contact plates into each chamber, threading machine screws through the plates and the actuator walls, and into the endcap. Silicone is poured into the chambers to create a thin sheet of rubber to seal that end.

Step 6: Vented plugs are fitted with high-pressure tubing, inserted into the open ends of the actuator body, and secured with silicone. The silicone seals the chambers from the inside and only allows air to flow through the tubing.

Step 7: Finally, the connector end cap is attached to the vented end of the actuator using a nylon machine screw, which is threaded through the end cap and the mount tabs on the protruding strain limiting layer (Fig. 5–Step 1).

The completed actuators are connected to a gas source using barbed tubing adapters on both chambers. The actuator can be connected in series with other inflatable components (phalanges) using the mount holes located in the end cap (Fig. 6).

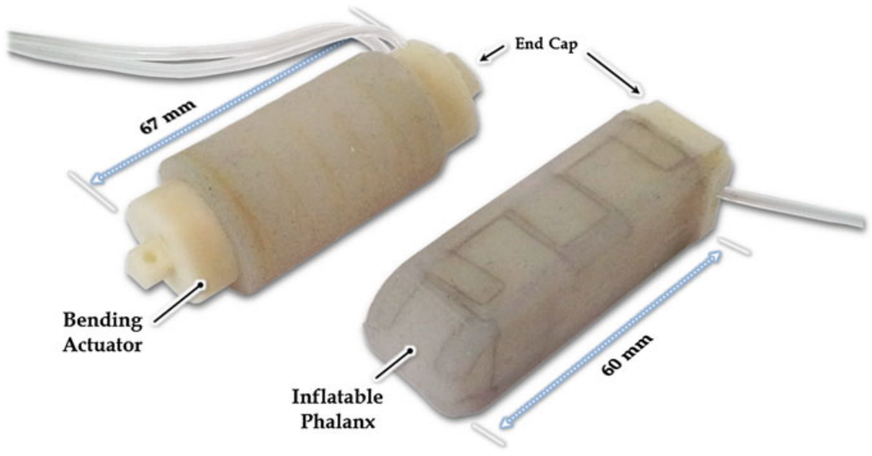


Fig. 6 Photos of the completed pneumatic phalanx and bending actuator

4.3 Pneumatic Grasp Assist Device Assembly

The proposed pneumatic grasp assist device is comprised of two SR fingers and a forearm brace. The brace, shown in Figs. 1 and 2, is comprised of three cuff plates which are padded with neoprene and fitted with hook-and-loop backed strips and fastening buckles. The SR finger mount on the main cuff plate is detachable, allowing easy modification of the SR finger device (swapping of SR finger types), which is useful for clinical applications where patients' grasp assist needs vary.

5 Experimental Validation

5.1 Semi-rigid Phalanx Stiffness

To validate the phalanx's stiffness against loads imparted during ADL grasping tasks, it is placed in a cantilever configuration and a range of fingertip loads are applied. Figure 7 shows the phalanx mounted horizontally in a test mount, in cantilever configuration. A contact bracket is placed over the phalanx tip so that weights can be suspended from the phalanx to apply the normal loads.

For each loading test, the phalanx is first inflated to maximum initial pressures of 0, 5, and 10 psi (while no load is applied) and scientific weights are suspended 5 mm from the phalanx tip. This load is increased from 0–750 g by adding weights to the suspended stack. Deformation of the phalanx is measured using five fiducial markers on the side of the phalanx, and the internal phalanx pressures are measured using a pressure gauge (ASDX-100PA, Honeywell Inc.) and recorded in MATLAB (Mathworks Inc.). A plot of phalanx deformation versus normal load is shown for

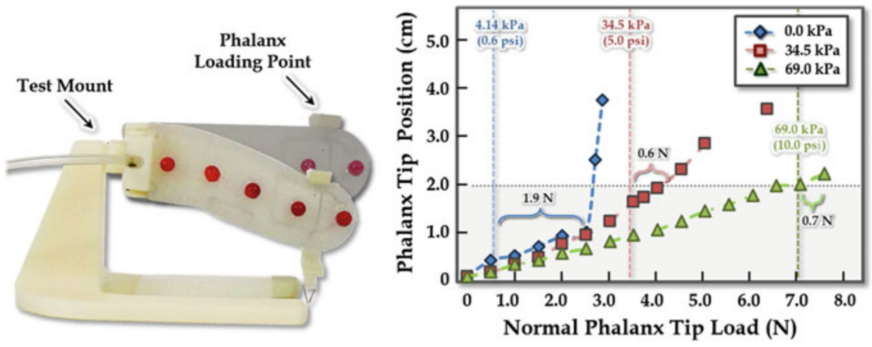


Fig. 7 Phalanx test setup and data: *Horizontal dashed line* at 2.0 cm represents deflection limit. *Dashed vertical lines* indicate estimated buckling load and brackets show error

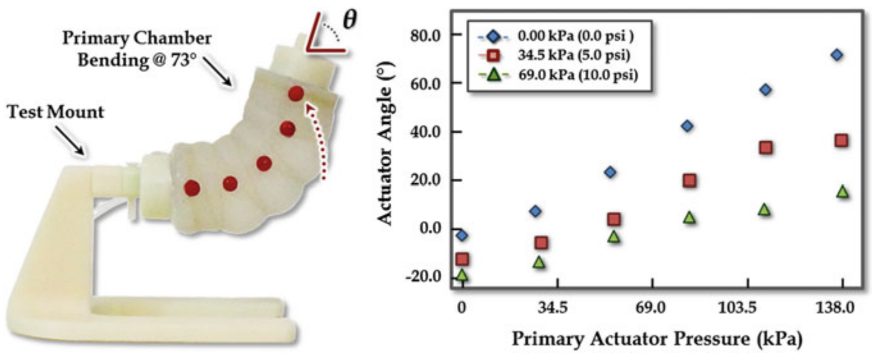


Fig. 8 Plot of actuator bending angle versus primary chamber pressure

each of the three initial pressures in Fig. 7. Critical buckling loads, predicted by Eq. 1 are shown as dashed, color-coded vertical lines.

5.2 *Characterizing Actuator Deformation*

The bending angle and stiffness of the bidirectional bending actuator are determined by the primary and antagonist chamber pressures and the loads applied. To characterize these mechanical properties, we tested the actuator in two conditions:

Condition 1- No Load: The primary actuator chamber is pressurized from atmosphere to 20 psi and the deformation (bending angle and position) are measured against internal pressure. The antagonist pressure chamber is pressurized at 0 psi (vent to atmosphere), 5, and 10 psi (Fig. 8).

Condition 2 - Stiffness Characterization: The antagonist chamber is pressurized to 5, 10, 15, and 20 psi, and then the primary chamber is pressurized so that the angle is at 0°. A 200 g mass is suspended from the actuator tip and the resulting deformation is recorded using fiducials on the actuator body.

The bending actuator stiffness characterization test setup is shown in Fig. 9, along with the resulting experimental data.

5.3 *Programming Motion Synergies*

To demonstrate programmable motion synergies, the bidirectional actuators in the grasp assist device are pre-stressed with different antagonist pressures, allowing the actuators to move at different rates with respect to the shared primary chamber pressure source. Figure 10 shows a successful grasp of an ADL object achieved using this method. The forearm brace and SR finger mount are positioned so that the SR

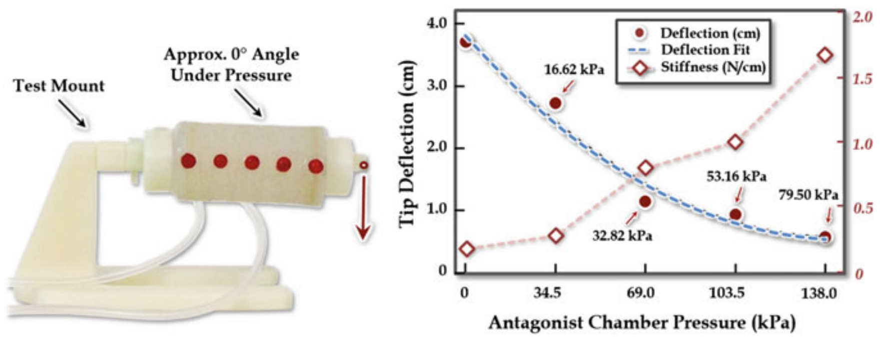


Fig. 9 Plot of actuator deflection and stiffness (force/deflection) versus chamber pressures. Primary chamber pressures are indicated next to deflection markers on the plot. Bending stiffness values (N/cm of tip deflection) are italicized and shown in *red* on *right* Y-axis

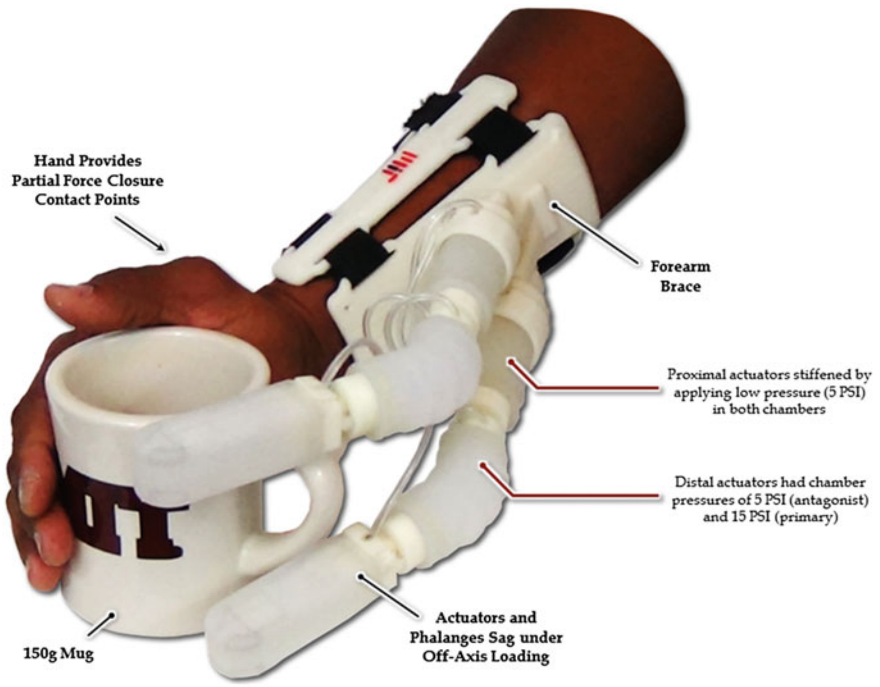

Hand Provides Partial Force Closure Contact Points
Forearm Brace
Proximal actuators stiffened by applying low pressure (5 PSI) in both chambers
Distal actuators had chamber pressures of 5 PSI (antagonist) and 15 PSI (primary)
Actuators and Phalanges Sag under Off-Axis Loading
150g Mug

Fig. 10 Photo of a mug grasped using the pneumatic SR grasp assist device

finger tips meet those of the human user, promoting better form closure on objects that require open-hand grasp postures (e.g. books, plates, etc.). The brace and SR finger mount positions can be modified for patient specific needs.

6 Results and Discussion

6.1 Phalanx Mechanical Stiffness

The simplified approximation of inflatable semi-rigid phalanx buckling resulted in buckling pressure calculations that matched the trends of the physical phenomenon but were not accurate. The average error between the actual and predicted buckling loads was 13.69 psi. This disparity is likely due to model simplicity.

The inflatable phalanx performed better than predicted, withstanding more than 7.35 N of normal force (750 g mass) without buckling (Fig. 7). This 7.35 N is much lower than the 175 N maximum exertion by the human thumb [1], but is adequate for grasping objects of ADL. Phalanx stiffness can be improved by increasing the internal pressure and/or using phalanges with stronger walls.

6.2 Pneumatic Bending Actuator Performance

The bending actuator exhibited a maximum of 87° range of motion in the no load and antagonist chamber vented condition. As the antagonist pressure increased, the range and rate of motion decreased significantly. This behavior lends itself to strategic programming of actuator motion rate to achieve certain grasp postures and even stiffness for certain types of ADL objects or tasks. The ability to modulate stiffness, as shown in Fig. 9, is particularly useful for tuning grasp strength when high forces and torques must be applied to the objects being used.

6.3 Pneumatic SR Grasp Assistance Experiments

The pneumatic SR grasp assist device experiment demonstrated that the bending actuators and inflatable phalanges were strong enough to apply contact forces required to manipulate ADL objects and that the ability to modulate finger motion can be leveraged to increase grasp quality. Also demonstrated was the desired passive underactuated behavior during grasp acquisition, where the pneumatic SR fingers conformed to the object using only one active power source and no low-level position control. The object grasp shown in Fig. 10 required greater stiffness in the distal actuators than the proximal ones, and this was achieved by applying higher antagonist pressures to those joints. Stiffness modulation also decreased gravity-induced lateral sag in the fingers by $\sim$70% (from 4.3 cm down to 1.3 cm).

7 Conclusion

This paper presents the design, analysis, and experimental validation of novel variable stiffness pneumatic bending actuators and inflatable semi-rigid phalanges for programming motion synergies in wearable robotic devices. Soft robot fabrication methods are presented and functional performance specifications, including motion range and stiffness, are modeled and experimentally validated. Grasp assist trials using the pneumatic SR grasp assist device demonstrate the ability to strategically modulate actuator motion to produce grasps suited to specific objects.

The modeling of the inflatable structures in this work is based primarily on simplified mechanical models due to the large, highly-nonlinear deformation of the elastic components and numerous strain limiting elements. Future research efforts will consider a more rigorous treatment of deformation models which predicts mechanical behavior based on material types and component design, and allows design optimization. Future research efforts will also consider implementing additional degrees of freedom for the pneumatic SR fingers (abduction–adduction) and will make use of soft wearable sensors and pressure regulators to create a closed-loop system that moves the SR fingers based upon user motion inputs.

Acknowledgements The authors gratefully acknowledge Dr. David Crandell at Spaulding Rehabilitation Hospital in Boston, MA for his expertise on the stroke rehabilitation and impaired hand function, and Prof. Rohan Abeyaratne in the MIT Department of Mechanical Engineering for his expertise on nonlinear elastic deformation mechanics.

References

1. An, K., Chao, E., Cooney, W., Linscheid, R.: Forces in the normal and abnormal hand. J. Orthop. Res. **3**, 202–211 (1985)
2. Balasubramanian, S., Klein, J., Burdet, E.: Robot-assisted rehabilitation of hand function. Curr. Opin. Neurol. **23**(6), 661–670 (2010)
3. Bishop-Moser, J., Krishnan, G., Kim, C., Kota, S.: Design of soft robotic actuators using fluid-filled fiber-reinforced elastomeric enclosures in parallel combinations. In: IEEE International Conference on Intelligent Robots and Systems, pp. 4264–4269 (2012)
4. Brown, E., et al.: Universal robotic gripper based on the jamming of granular material. Proc. Natl. Acad. Sci. **107**(44), 18809–18814 (2010)
5. Canela, M., del Ama, A.J., Pons, J.L.: Design of a pediatric exoskeleton for the rehabilitation of the physical disabilities caused by cerebral palsy. Biosyst. Biorobot. **1**, 255–258 (2013)
6. Cempini, M., Cortese, M., Vitiello, N.: A powered finger-thumb wearable hand exoskeleton with self-aligning joint axes. IEEE Trans. Mechatron. **20**(32), 705–716 (2014)
7. Connelly L. et al.: Use of a pneumatic glove for hand rehabilitation following stroke. In: IEEE International Conference on Engineering in Medicine and Biology Society, pp. 2434–2437 (2009)
8. Deimel, R., Brock, O.: A compliant hand based on a novel actuator. In: IEEE International Conference on Robotics and Auto, Karlsruhe, Germany, pp. 2047–2053 (2013)
9. Dollar, A.M., Howe, R.D.: The highly adaptive SDM hand: design and performance evaluation. Int. J. Robot. Res. **29**(5), 585–597 (2010)
10. Galloway, K.C., Polygerinos, P., Walsh, C., Wood, R.: Mechanically programmable bend radius for fiber-reinforced soft actuators. In: IEEE International Conference on Advanced Robotics, Montevideo, Uruguay, pp. 1–6 (2013)

11. Graf, C.: The Lawton instrumental activities of daily living scale. Am. J. Nurs. **108**(4), 52–62 (2008)
12. Heo, P., et al.: Current hand exoskeleton technologies for rehabilitation and assistive engineering. Int. J. Precis. Eng. Manuf. **13**(5), 807–824 (2012)
13. Ilievski, F., Mazzeo, A.D., Shepherd, R.F., Chen, X., Whitesides, G.M.: Soft robotics for chemists. Angew. Chem. **123**, 1930–1935 (2011)
14. Katz, S.: Assessing self-maintenance: activities of daily living, mobility, and instrumental activities of daily living. J. Am. Geriatr. Soc. **31**(12), 721–727 (1983)
15. Koeneman, E., Schultz, R., Wolf, S., Herring, D., Koeneman, J.: A pneumatic muscle hand therapy device. In: Proceedings of the IEEE International Conference on Engineering in Medicine and Biology Society, pp. 2711–2713 (2004)
16. Krebs, H.I., Hogan, N., Aisen, M.L., Volpe, B.T.: Robot-Aided neurorehabilitation. IEEE Trans. Rehabil. Eng. **6**(1), 75–87 (1998)
17. Kutner, N., et al.: Quality-of-life change associated with robotic-assisted therapy to improve hand motor function in patients with subacute stroke: a randomized clinical trial. Phys. Ther. **90**, 493–504 (2010)
18. Kwakkel, G., Kollen, B.J., Krebs, H.I.: Effects of robot-assisted therapy on upper limb recovery after stroke: a systematic review. Neurorehabil. Neural Repair **22**(2), 111–121 (2008)
19. Llorens-Bonilla, B., Parietti, F., Asada, H.: Demonstration-based control of supernumerary robotic limbs. In: Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems, pp. 3936–3942 (2012)
20. McNeil, J.: Americans with disabilities: household economic studies, United States Census, pp. 70–73 (2011)
21. Noritsugu, T., Yamamoto, H., Sasakil, D., Takaiwa, M.: Wearable power assist device for hand grasping using pneumatic artificial rubber muscle. In: SICE 2004 Annual Conference, IEEE. **1**, 420–425 (2004)
22. Odhner, L., et al.: A compliant, underactuated hand for robust manipulation. Int. J. Robot. Res. **33**(5), 736–752 (2014)
23. Polygerinos, P. et al.: Towards a soft pneumatic glove for hand rehabilitation. In: IEEE International Conference of Robotics and Automation, Tokyo, Japan (2013)
24. Park, Y.-L. et al.: Design and control of a bio-inspired soft wearable robotic device for ankle-foot rehabilitation. Bioinspir. Biomim., **9**(1) (2014)
25. Parietti, F., Asada, H.: Bracing the human body with supernumerary robotic limbs for physical assistance and load reduction. In: Proceedings of the IEEE International Conference on Robotics and Automation, Hong Kong, China (2014)
26. Pons, J.L.: Rehabilitation exoskeletal robotics. Promise Emerg. Field IEEE Eng. Med. Biol. Mag. **29**(3), 57–63 (2010)
27. Sasaki D., Noritsugu T., Takaiwa M., Yamamoto H.: Wearable power assist device for hand grasping using pneumatic artificial rubber muscle. In: Proceedings of the IEEE International Workshop ROMAN, pp. 655–660
28. Shapiro, Y., Wolf, A., Gabor, K.: Bi-bellows: pneumatic bending actuator. Sensor Actuators A Phys. **167**, 484–494 (2011)
29. Suzumori, K.: Elastic materials producing compliant robots. Robot. Auton. Syst. **18**, 135–140 (1996)
30. Takahashi, C.D., Der-Yeghiaian, L., Le, V., Motiwala, R.R., Cramer, S.C.: Robot-based hand motor therapy after stroke. Brain **131**(2), 425–437 (2008)
31. Wu, F.Y., Asada, H.H.: Bio-Artificial Synergies for Grasp Posture Control of Supernumerary Robotic Fingers, Robotics: Science and Systems (2014)
32. Wu, F.Y., Asada, H.H.: Hold-and-manipulate with a single hand being assisted by wearable extra fingers. In: IEEE International Conference on Robotics and Automation, Seattle, Washington (2015)
33. Xing, K., Xu, Q., Wang, Y.: Design of a wearable rehabilitation robotic hand actuated by pneumatic artificial muscles. In: Proceedings of the 7th Asian Control Conference, Hong Kong, China, pp. 740–744 (2009)

Reproducing Expert-Like Motion in Deformable Environments Using Active Learning and IOC

Calder Phillips-Grafflin and Dmitry Berenson

1 Introduction

Manipulation of deformable objects, and in deformable environments, is an important area of research as deformable objects are common in domestic, industrial, and medical environments. Unlike manipulation in rigid environments, where collisions are forbidden, deformable environments allow, and often require, collisions between a robot and deformable objects. However, modeling deformable objects is a difficult problem; models must not only capture the geometry (undeformed and deformed) of objects (itself a very difficult problem), but should also capture the sensitivity of the object. This qualitative aspect is critical for deformable environments, as it allows a motion planner to distinguish between multiple objects with similar physical properties but with different qualitative characteristics. An important example of this occurs in surgical robotics; while multiple organs and tissues may have similar physical properties, some parts of the body are significantly more sensitive than others. Without accounting for sensitivity, motion planners can produce paths that could cause unnecessary injury.

The motion planning methods introduced in our previous work [16] use a voxel-based representation of deformable objects in which each voxel has two parameters. The first parameter, *deformability*, captures physical properties of the rigidity of the material. The second parameter, *sensitivity*, captures the qualitative significance of

This work is supported in part by the Office of Naval Research under Grant N00014-13-1-0735 and by the National Science Foundation under Grant IIS-1317462.

C. Phillips-Grafflin (✉)
Worcester Polytechnic Institute, 100 Institute Rd, Worcester, MA 01609, USA
e-mail: cnphillipsgraffl@wpi.edu

D. Berenson
University of Michigan, 1301 Beal Avenue, Ann Arbor, MI 48109, USA
e-mail: berenson@eecs.umich.edu

A. Bicchi and W. Burgard (eds.), *Robotics Research*, Springer Proceedings in Advanced Robotics 2, DOI 10.1007/978-3-319-51532-8_14

deforming the object. Together, these parameters are used in a cost function that provides a cost of deformation that can be used in cost-aware motion planners.

While the deformability parameters are directly related to material properties, setting the sensitivity parameters is more difficult, as they capture a range of object characteristics. Setting them by hand is time-consuming and error-prone, as incorrect sensitivity values can produce unwanted planner behavior. More problematically, setting these parameters for practical environments requires both domain knowledge and the ability to mathematically represent that knowledge such that the planner will perform well. Instead, we propose a framework for automatically learning and validating these parameters from expert demonstrations. For example, a surgeon can demonstrate the optimal path for inserting a probe, and we can use this demonstration to find the sensitivity values of organs around the path.

Our framework consists of three parts: (1) Automatic generation of demonstration tasks that prompt the user to provide informative demonstrations through a novel active learning process; (2) Recovery of object sensitivity values using *Path Integral Inverse Reinforcement Learning* (PIIRL) Inverse Optimal Control (IOC) techniques [10]; and (3) Reproduction of the demonstrated behavior using the RRT* asymptotically-optimal motion planner [12] with a key modification that allows us to check for punctures of deformable objects.

This approach offers two main advantages over existing similar techniques. First, by using sampling-based techniques for IOC that avoid the need to solve the forward problem as well as sampling-based asymptotically-optimal planners, our framework is applicable to higher-dimensional problems than approaches such as LEARCH [17], which are limited by the need to repeatedly compute optimal paths to recover the cost function. Second, our proposed method for automatically generating demonstration tasks for experts to perform reduces the number of demonstration tasks needed to capture the desired behavior and removes the need for domain knowledge to generate these tasks by hand. Finally, to our knowledge, IOC has never before been applied to the problem of learning deformable object parameters.

In our experiments in simulated and physical test environments we show that, despite the limitations inherent in asymptotically-optimal sampling-based planning, the recovered sensitivity parameters allow motion planners to reliably reproduce behavior demonstrated by expert users. We also present experiments which show the generalization capabilities of our method.

2 Related Work

Extensive work on the modeling and representation of deformable objects has been done, primarily from the perspectives of computer graphics [6] and medicine [2]. In recent years, this work has been adopted by the robotics field to enable the manipulation of real-world deformable objects such as clothing, rope, food, and human tissue [8]. A wide range of simulation-based models for deformable objects are available, most of which are meshed models based on Mass-Spring (M-S) [6] and

Finite-Element (FEM) [7, 14], but mesh-less models [5, 13, 16] have also been proposed. Our model replaces the need for physical simulation with a cost function based on the volume of intersection between voxelized deformable and rigid objects [16]. While this approach cannot capture moving objects, and can only approximate the true deformation, it is extremely efficient to compute in comparison to simulation-based methods, and thus ideal for use inside a motion planner. Notably, because our approach provides a cost function that accounts for both object deformation and sensitivity, it produces plans that minimize deformation and preferentially deform or avoid objects based on their sensitivity.

Inverse Optimal Control (IOC) is the problem of recovering the cost or reward function being optimized by a trajectory or policy. Introduced by Kalman [11] and applied to robotics by Ng et al. [15], several different formulations of the IOC problem and algorithms to address it have been proposed, covering both continuous and discrete state spaces [15]. Earlier approaches to the IOC problem, such as apprenticeship learning, require that the forward problem be solved in addition to computing optimal weights [1, 17]. More recent approaches, based on the maximum entropy principle, replace the need for solving the forward problem by using sample trajectories around the demonstration [19].

The IOC approach we use, *Path Integral Inverse Reinforcement Learning* (PIIRL) samples around the demonstration instead of solving the forward problem [10]. In the PIIRL formulation, a series of locally-optimal demonstration trajectories are gathered from the user(s). For each of these demonstrations, a set of sample trajectories around the demonstration is generated; note that these samples are assumed to be sub-optimal relative to their demonstration. For all demonstrations and all samples, user-specified features are evaluated, and the weights associated with these features are then recovered via a convex optimization problem that attempts to maximize the margin between the features of the demonstrations and the features of the samples.

3 Problem Statement

Let τ represent the path of a rigid object (i.e. the robot) through an environment composed of n deformable objects $E = O_1, O_2, \dots, O_n$. Representing τ with a discrete sequence of configurations, we assume the cost of executing τ is a function of the form $C(\tau) = \sum_{k=1}^{|\tau|} \sum_{i=1}^{n} D_i S_i V_i(\tau_k)$, where $V_i(\tau_k)$ is the volume of deformation of O_i that results from placing the rigid object at the kth configuration of path τ, D_i is the deformability of O_i, and S_i is the sensitivity of O_i. We focus on learning the S_i parameters, so we assume $D_i = 1\ \forall i$, though our methods work with any known D. Note that while sensitivity parameters S can be set per-voxel in our representation, we simplify the problem of recovering sensitivities by assuming that each object has uniform sensitivity.

S represents the ground-truth sensitivities of the objects. We seek to generate a set of learned parameters $\hat{S}$ from a set of demonstrations, such that these $\hat{S}$ can be used in a motion planner to produce similar behavior to the demonstrations. Obtaining the true S from demonstration is not possible in general, as a demonstration can, at best, encode only the ratios between different elements of S and not their magnitudes. Thus it is not meaningful to compare S to $\hat{S}$ directly. A more informative comparison is how well a planner imitates demonstrated behavior when planning with $\hat{S}$. Thus we evaluate our method in terms of the cost of the path produced by our framework. Therefore the quality of $\hat{S}$ relative to the ground truth is evaluated as $E(\hat{S}, S) = |C_S(\tau_d) - C_S(\tau_{planned}(\hat{S}))|$, where τ_d is a path demonstrated for a given task, $\tau_{planned}(\hat{S})$ is a path planned for the same task using the sensitivities $\hat{S}$, and the cost function $C_S(\cdot)$ is evaluated using the ground-truth sensitivities S.

4 Methods

We have developed a framework for recovering sensitivity parameters for deformable objects, as illustrated in Fig. 1. Below we describe each of the four components in detail.

4.1 Capturing Demonstrations

Like all IOC problems, our approach requires demonstrations. In our case, demonstrations are captured in a simulation environment using a physics simulator to simulate deformable objects. Our demonstration task consists of inserting a cylindrical probe between deformable objects to reach target points distributed across the environment, as illustrated in Fig. 2. The user attempts to minimize contact with more sensitive objects (shown in yellow and green) compared to less sensitive objects (shown in blue). We record the demonstration trajectory along with the features of that trajectory, which are the total amounts of deformation of each object. While outwardly simple, the problem of probe and needle insertion between deformable objects such as this is common in medical tasks [2] and a subject of previous research

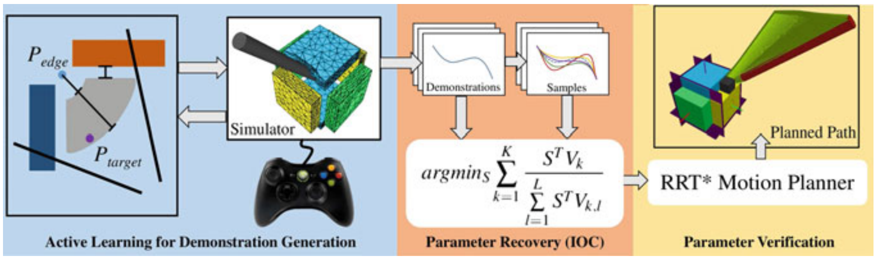


Fig. 1 Diagram of the three stages and main components of our framework

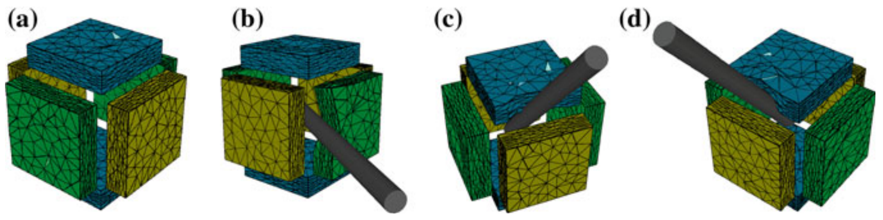
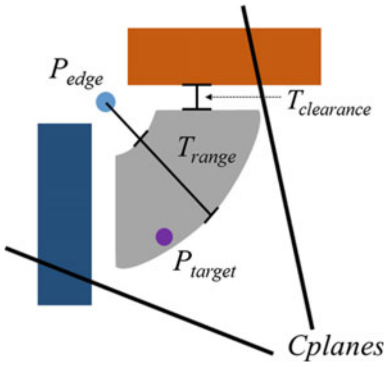


Fig. 2 Example demonstration tasks for our 6-object test environment, shown with the probe reaching the target. **a** Low sensitivity objects L_1, L_2 (*blue*), medium sensitivity objects M_1, M_2 (*green*), and high sensitivity objects H_1, H_2 (*yellow*). **b–d** Goal configurations for three automatically generated tasks

Fig. 3 Our automatic demonstration task generator



in robot motion [2, 13], however, none of this work has explored learning qualitative properties of deformable objects to determine higher-level behavior. In addition to capturing demonstrations, we use this simulation environment to compute feature vectors for demonstration and sample paths.

Each demonstration we capture can be parametrized as a *demonstration task* by a starting pose of the probe P_{start}, a target point P_{target} the user must touch with the probe tip, and a set of "collision planes" $Cplanes$, hyperplanes that constrain the motion of the probe. As shown in Fig. 3, the hyperplanes approximate a funnel that guides the user towards the target point and restricts which objects the user can contact with the probe. These hyperplanes are added to constrain the user to producing demonstrations that capture the relative difference in sensitivity between the accessible objects. In our experience, without the hyperplanes users sometimes produce demonstrations that deform only the globally least-sensitive object(s) instead of capturing sensitivity relationships between neighboring objects.

While we attempt to capture optimal demonstrations, in practice users may provide slightly sub-optimal demonstrations. We attempt to correct for this using a local optimizer that optimizes each demonstration. This method generates a set of random sample trajectories around the demonstration trajectory and replaces the demonstration trajectory with any of the random samples with strictly dominating deformation (i.e. the random sample deforms all objects less than or equal to the demonstration).

4.2 Active Learning

We can capture demonstrations and compute features for demonstrations and samples needed for PIIRL, however, this leaves two problems to address: how to generate demonstration tasks for the user to complete, and how many demonstrations must be collected. Clearly, the accuracy of recovered sensitivity values depends on the quality of the demonstrations provided. For example, if an object has zero feature values in both demonstrations and the trajectory samples around the demonstrations used by PIIRL, we cannot recover a meaningful sensitivity value for the object; e.g. if all demonstrations entered through the forward half of our cube environment, no features would be available for objects on the reverse. A different, but equally problematic, issue occurs when features have been collected for every object, but the demonstrations are "unconnected"; for example, in an environment $E = \{O_1, O_2, O_3, O_4\}$, if features have been collected for demonstrations between O_1, O_2 and O_3, O_4, but not for O_2, O_3, the optimizer cannot determine if O_1 and O_2 are more or less sensitive that O_3 and O_4. Thus, we need to ensure that sufficient demonstrations have been collected.

Algorithm 1 Demonstration task collection algorithm

```
procedure CollectDemonstrations($A$)
    $G \leftarrow \{\emptyset, \emptyset\}$
    $O_1 \leftarrow argmax_{o \in E}$ degree$(A(o))$
    $O_2 \leftarrow argmax_{o \in \text{neighbors}(A(O_1))}$degree $(A(o))$
    $G \leftarrow G \cup$ CollectSingleDemonstration$(O_1, O_2)$
    while $\{o \in E | o \notin G_v$, degree$(A(o)) > 0\} \neq \emptyset$ do
        $O_1 \leftarrow argmax_{\{o \in G_v | \text{neighbors}(A(o)) \setminus G_v \neq \emptyset\}}$ depth$(G(o))$
        $O_2 \leftarrow argmax_{\{o \in \text{neighbors}(A(O_1)) \setminus G_v\}}$ degree$(A(o))$
        $G \leftarrow G \cup$ CollectSingleDemonstration$(O_1, O_2)$
        $G \leftarrow$ EnsureRanking$(G)$
    return $G$
procedure EnsureRanking($(G)$)
    for $O_1 \in G_v$ do
        for $\{O_2 \in G_v |$ depth$(G(O_2)) \geq$ depth$(G(O_1))\}$ do
            if NoDirectedPathExists$(O_1, O_2)$ then
                if DirectlyComparable$(O_1, O_2)$ then
                    $G \leftarrow G \cup$ CollectSingleDemonstration$(O_1, O_2)$
                    return EnsureRanking$(G)$
    return $G$
procedure CollectSingleDemonstration($O_1, O_2$)
    $P_{target}, P_{edge}, C_{planes} \leftarrow$ GenerateDemonstrationTask$(O_1, O_2, T_{clearance}, T_{range})$
    $D_v, D_e \leftarrow$ GetDemonstrationFromUser$(P_{target}, P_{edge}, C_{planes})$
    return $(D_v, D_e)$
```

The conservative solution is to require a demonstration for every pair of adjacent objects, however, this can result in a large number of demonstrations. For our test environment shown in Fig. 2, 12 demonstrations would be required to capture the relationship between every adjacent pair. We seek to reduce the number of demonstrations required.

Simply collecting demonstrations such that we observe a non-zero feature for each object is insufficient for accurate parameter recovery, rather, we must ensure that the demonstrations collected form a *ranking* of the objects in terms of sensitivity; i.e. that for objects $O_1, O_2 \in E$, $rank(O_1)$ is either less than, equal to, or greater than $rank(O_2)$ if the objects are comparable. Rankings are derived from demonstrations collected between adjacent objects; the preferentially-deformed object receives a lower ranking than the preferentially-avoided object. Rankings are not comparable in certain cases, such as between objects on the opposing faces of our test environment, in which it impossible to perform a demonstration between the two objects, and they cannot be ranked via a combination of other demonstrations.

Demonstrations are collected using Algorithm 1, which takes A, the set of object adjacencies in E, and iteratively collects demonstrations until there are no more useful demonstrations to perform. This algorithm captures preference relationships between objects by building a directed graph G. The nodes in G represent objects in the environment and the directed edges point from the less-sensitive object to the more-sensitive one. Initially, G contains no nodes or edges, and each demonstration adds an edge and 0, 1, or 2 nodes. The key to the algorithm is determining which demonstration (and thus which edge) should be queried next.

The algorithm uses the structure of G at the current time as well as a heuristic to decide which demonstration to query next. If the ranking between all objects in G is known, then the algorithm selects a new object to add to G (via a demonstration involving that object and one already in G). After adding a new object, the algorithm queries demonstrations until the ranking of all objects in the graph is again established (this is done in the ENSURERANKING function). It then selects a new object to add, and so on, until no more objects can be added.

At each step where objects or edges are selected, we choose the object or edge based on connectivity heuristics. For new objects (i.e. those not already in G), we prefer those that are adjacent to as many other objects as possible. When picking objects already in G for a new edge, we prefer objects that have a higher "depth". Here depth(n) is the length of the longest directed path in G which ends at n. These heuristics bias the algorithm to create long chains of edges where possible, which is clearly beneficial for forming a ranked list; e.g. $rank(O_1) < rank(O_2) < rank(O_3) < rank(O_4)$ is a chain of three edges which gives a complete ranking of four objects.

Algorithm 1 is not guaranteed to produce the minimal set of demonstrations because it cannot foresee the results of future demonstrations. It frequently collects demonstrations early on that prove to be unnecessary in the final set of demonstrations. In pathological environments, Algorithm 1 may be forced to collect all possible demonstrations. However, in practice, we show that it reduces the number of demonstrations without significant impact on the recovered sensitivity parameters.

Algorithm 2 Demonstration task generation algorithm

```
procedure GenerateDemonstrationTask($O_1$, $O_2$, $T_{clearance}$, $T_{range}$)
    $P_{edge} \leftarrow$ GetEdgePointBetweenObjects($O_1$, $O_2$)
    $P_{target} \leftarrow \emptyset$
    while $P_{target} = \emptyset$ do
        $P_{sampled} \leftarrow$ SampleInRange($P_{edge}$, $T_{range}$)
        if $clearance(P_{sampled}) < T_{clearance}$ then
            $P_{target} \leftarrow P_{sampled}$
    $Cplanes \leftarrow$ GenerateCollisionPlanes($P_{edge}$, $P_{target}$)
    return $P_{target}$, $P_{edge}$, $Cplanes$
```

For each demonstration requested by Algorithm 1, we generate a new task using Algorithm 2. This algorithm is given a pair of target objects O_1, O_2, a target clearance $T_{clearance}$, and a target depth range T_{range}. First, the algorithm selects an "edge point", P_{edge} by randomly selecting a point on the medial axis between the two target objects. Using the edge point, the algorithm randomly samples nearby points T_{range} away from the edge point to select one that is "inside"[1] the environment and also at least $T_{clearance}$ away from an object, which it returns as P_{target}, the target point. Finally, a set of "collision planes" are generated to restrict the user's demonstration to the desired area. The parameter T_{range} ensures that the user must insert the probe sufficiently to cause deformations. Similarly, the parameter $T_{clearance}$ controls how close to an object the target point can be, and can be used to ensure that the target point itself is not in contact with an object (see Fig. 3).

4.3 Parameter Recovery

Our approach to motion planning for deformable objects, introduced in [16], uses a "cost of deformation" to enable any motion planner that accounts for cost to produce plans that minimize deformation. We can frame the problem of imitating demonstration behavior as the problem of inferring the sensitivity parameters used to produce the demonstration. Assuming that the demonstration is optimal, this is the well-established problem of Inverse Optimal Control (IOC).

Using the PIIRL formulation of IOC, the cost function consists of a series of features $V = V_1, V_2, \ldots, V_n$ (in our case these are the amounts of deformation of each of the n objects) with corresponding sensitivities $S = S_1, S_2, \ldots, S_n$, such that

[1]To determine which points are "inside" the environment, we compute a "local maxima map" using the Signed Distance Field (SDF) of the environment. For each point in the SDF, we follow the gradient away from obstacles and record the location the gradient becomes zero (i.e. the local distance maxima). Points "inside" the environment have corresponding local maxima inside the bounds of the SDF, while points "outside" have local maxima corresponding to the bounds of the SDF. Intuitively, "inside" points have finite-distance local maxima reachable via the gradient, while for "outside" points, the local maxima are undefined.

the total cost of a configuration $C = \sum_{i=1}^{n} V_i S_i$, where the V_i can be computed using our physics simulator, but the optimal set of sensitivities S^* is unknown.

To find the best estimate of the optimal set of sensitivities $\hat{S}$, PIIRL requires a set of sample paths around each demonstration. Because the demonstrations are assumed to be locally optimal, all samples around a demonstration will be sub-optimal w.r.t. the unknown cost function. For K demonstrations and L samples for each demonstration, the optimal weights are obtained using the following minimization problem (a similar form of the minimization problem used in [10]), where V_k are the feature values for demonstration k, and $V_{k,l}$ are the feature values for sample l of demonstration k:

$$\hat{S} = argmin_S \sum_{k=1}^{K} \frac{S^T V_k}{\sum_{l=1}^{L} S^T V_{k,l}} \tag{1}$$

This minimization finds the sensitivity values $\hat{S}$ that maximize the margin between the cost of the demonstrations and the costs of their samples. Note that in our problem, $S > 0$ and sample feature values $V_{k,l} \neq 0$, as all sensitivity values must be greater than zero and $V_{k,l} = 0$ implies $V_k = 0$ (since samples must be sub-optimal relative to their demonstrations). $V_k = 0$ implies that the demonstration k captures no information about any object and thus can be removed from the optimization so this condition will not occur. Since this minimization problem is convex, we can use standard convex optimization solvers to find optimal weights. Unlike previous work such as LEARCH [17], PIIRL does not rely on the specific configurations the demonstration path traverses; rather, only the corresponding feature values must be locally optimal in our cost function [9]. This makes it tractable to learn cost functions in high-dimensional spaces.

4.4 Recovered Parameter Verification

Once sensitivity values $\hat{S}$ have been recovered for each object in our test environments, we must verify that the recovered values allow our motion planner to imitate the behavior of the expert demonstrations. We attempt to perform each demonstration task using an optimal motion planner and comparing the planned path $\tau_{planned}(\hat{S})$ with the demonstration τ_d in terms of the true cost function $C_S(\cdot)$ using the ground truth sensitivity values S. In our previous work, we used the T-RRT and GradienT-RRT planners to efficiently produce paths in high-dimensional spaces [16]; however, since these planners have no optimality guarantees, they are unsuitable for parameter verification. Instead, we use the asymptotically-optimal RRT* planner [12] with our deformation cost function. While we could use deformations measured via a physics simulator to compute cost during planning, our voxel-based deformation cost function is significantly faster, more stable, and detects object punctures and separation. To accurately mimic the demonstration tasks, the RRT* planner is provided with the same task-space target point to reach with the probe tip, rather than a goal configuration of the probe. Feasible configurations touching this target point can be sampled, and RRT* attempts to connect the tree to these goal states. As RRT* runs, it improves the path by reducing the deformation cost of the path and by sampling and connecting to new, lower-cost goal states. Note that while RRT* is *asymptotically* optimal, for finite time it will not return *the optimal* path, so we expect paths reproduced with RRT* may be slightly higher-cost than their corresponding expert demonstrations, but should exhibit the same preferential deformation demonstrated by the expert.

In addition to integrating our existing cost function with RRT*, we have significantly improved the quality of planned paths by adding puncture detection to prevent paths from puncturing or cutting deformable objects. Puncture and cut detection is essential to planner performance; without it, planners can produce low-cost paths that pass directly though deformable objects. To prevent punctures and cuts, we check every extension of the tree in RRT* for puncture using an incremental variant of the algorithm introduced by Chen et al. [3] for computing topological invariants on voxel grids. The original algorithm extracts the surface vertices from the voxel grid, and computes the connectivity of each surface vertex. Each surface vertex can be connected to between one and six neighboring surface vertices; let $M1$ be the total number of surface vertices with one connected neighbor, $M2$ the total with two neighbors, and so on. From these totals, Chen et al. prove that the number of holes in the voxel grid is $n_{holes} = 1 + ((M5 + (2 * M6) - M3)/8)$.

Thus, checking for punctures can be implemented by removing the swept volume of the path of the probe from the voxel-based model of deformable objects used for motion planning, and then computing the number of holes to ensure that no new holes have been created by the path. Additionally, to prevent objects from being completely cut apart by the path, the overall connectivity of the surface voxels corresponding to each object are computed; if the surface vertices for an object form multiple disconnected groups, then the object has been cut apart by the path.

To efficiently perform these checks during the planning process, we incrementally check for punctures with each extension and rewiring step of RRT* (see Fig. 4). For testing a new edge from configuration q_1 to configuration q_2 the process is as follows: (1) retrieve the stored object surfaces corresponding to q_1, (2) update the object surfaces with the swept volume from q_1 to q_2, (3) compute the number of

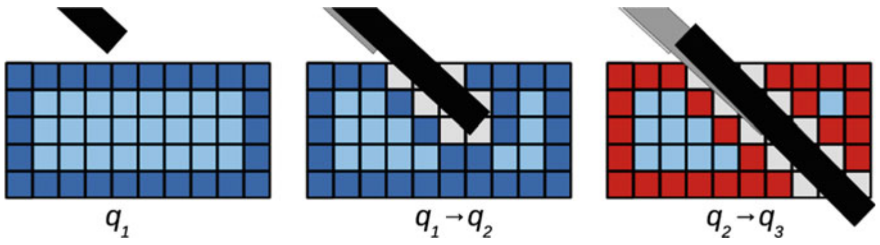


Fig. 4 Illustration of puncture checking for an extension from configuration q_1 to q_3. As the surfaces are no longer connected (*red*), puncture has occurred and the $q_2 \rightarrow q_3$ motion is invalid

holes in each object surface (check for puncture), (4) compute the connectivity of each object surface (check for cuts), and (5) if no holes or cuts are encountered, store the updated surfaces corresponding to q_2. For every such check, we are effectively checking the entire path from the start configuration q_{start} to q_2 for punctures and cuts.

5 Results

We present results of testing our framework in a 3D simulated environment (5DoF probe insertion task) and in a physical planar environment (3DoF rigid object navigation task) using an industrial robot. We use the Bullet physics simulator [4] to provide an environment for capturing demonstrations and computing features, and the Open Motion Planning Library (OMPL) [18] to provide the RRT* planner used to verify the recovered sensitivity values. We show that our methods accurately recover sensitivity values that allow planners to imitate expert demonstrations. We also report on how the algorithm generalizes to a new task, where an obstacle is introduced into the environment, and report on the use of active learning for reducing the number of demonstrations required. Ideally, we would compare the performance of our framework with existing approaches such as LEARCH [17], however, these approaches require computing the true optimal path to perform IOC, which is intractable in the 5DoF probe insertion task.

5.1 Recovered Behavior

We first demonstrate the performance of our framework in the 3D simulated environment without using the automatic demonstration task generator, and show that our demonstration capture environment and parameter recovery process produce acceptable object sensitivity values. Using our RRT* planner, we show that the recovered sensitivities produce paths that imitate the expert demonstrations.

The test environment, as shown in Fig. 2, consists of six deformable objects forming the faces of a hollow cube. These objects form three classes; each pair of opposing faces has the same sensitivity assigned, with the lowest sensitivity (L_1, L_2) shown in blue, an intermediate sensitivity (M_1, M_2) shown in green, and high sensitivity (H_1, H_2) shown in yellow. For testing purposes, the "true" sensitivity values of these objects are set as $L_1, L_2 = 0.2$, $M_1, M_2 = 0.4$, $H_1, H_2 = 0.8$. We use the true values to evaluate the quality of paths planned with the recovered sensitivity values, but they are unknown to our IOC method.

Using the conservative approach discussed in Sect. 4.2, 12 demonstrations were performed, one for each pair of adjacent objects. Several examples of these demonstrations can be seen in Figs. 2 and 5. While time-consuming, this approach ensures that sufficient demonstrations have been collected to capture the desired behavior. In

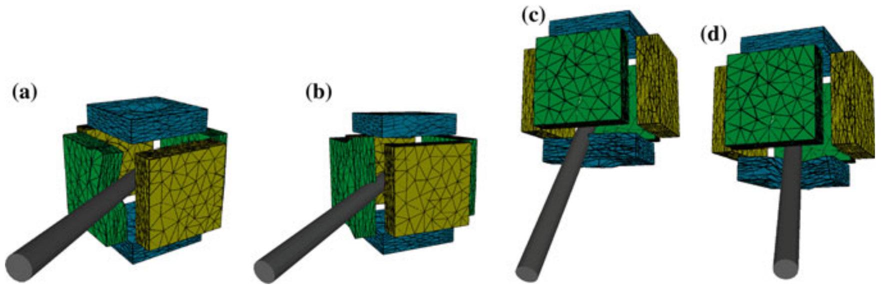


Fig. 5 Examples of goal configurations from demonstrations (**a**, **c**) and corresponding goals of paths planned using recovered sensitivity values (**b**, **d**). Full paths are not shown for clarity

these demonstrations, lower-sensitivity objects were preferentially deformed instead of higher-sensitivity objects.

Using the set of 12 demonstrations, we recovered the object sensitivity parameters using our parameter recovery process. We generated a set of 100 sample paths around each demonstration using a multivariate gaussian distribution using the process described in [9], which produces smooth noisy path samples around an initial path. Features for all demonstrations and samples were computed by executing paths in the demonstration capture environment, and all feature values were normalized relative to the highest feature value. Using the PIIRL formulation of IOC, the optimal weights were recovered using the convex optimization problem in Eq. (1); we used the function minimization tools in MATLAB to perform this optimization. For optimization, the lower bound of possible weight values was 0.1, and the upper bound was 1000, with the weights initialized to 500. The recovered sensitivity values were $L_1 = 0.10004$, $L_2 = 0.10092$, $M_1 = 2.8523$, $M_2 = 8.5683$, $H_1 = 958.92$, $H_2 = 999.51$. Note that both high sensitivity objects (H_1 and H_2) were avoided in all demonstrations, and thus received maximum weights in the optimization. Again, recovery of the true sensitivities is impossible and we must evaluate our method in terms of the cost of the path planned using the recovered sensitivities.

5.1.1 Recovered Parameter Verification

Using the object sensitivity parameters recovered using PIIRL, we planned for all 12 demonstration tasks using RRT*. Table 1 compares the demonstrations with results for planning times of 30 and 60 min, with 30 and 15 trials of each, respectively. Figure 5 shows examples of demonstrated paths compared with paths produced by RRT*. As shown in the table, paths produced using the recovered parameters imitate the behavior of the demonstrations by deforming the same objects with similar amounts of deformation except for two demonstrations (namely 9 and 12) for which the planner found a path superior to the original demonstration. Note that due to the difficulty of the planning problem and the finite planning time for RRT*, we do

Table 1 Comparison between demonstrated behavior and paths planned using object sensitivity values recovered from 12 demonstrations between each pair of adjacent objects. Costs reported (mean [std.dev.]) are the integral of volume change multiplied by the true object sensitivity values, separated by class of object (L = low-sensitivity, including objects L_1 and L_2, M = medium-sensitivity, including objects M_1 and M_2, H = high-sensitivity, including objects H_1 and H_2)

	Demonstrated			Recovered (30 plans, 30 min/plan)			Recovered (15 plans, 60 min/plan)		
	L	M	H	L	M	H	L	M	H
1	5.61	0.0	0.0	12.14 [4.68]	0.0 [0.0]	0.0 [0.0]	10.81 [1.55]	0.0 [0.0]	0.0 [0.0]
2	7.14	0.0	0.0	12.35 [2.7]	0.0 [0.0]	0.0 [0.0]	11.57 [2.82]	0.0 [0.0]	0.0 [0.0]
3	4.87	0.0	0.0	10.65 [3.71]	0.16 [0.82]	0.0 [0.0]	9.82 [2.84]	0.0 [0.0]	0.0 [0.0]
4	5.30	0.0	0.0	10.27 [2.75]	0.07 [0.38]	0.01 [0.03]	11.06 [2.22]	0.0 [0.0]	0.0 [0.0]
5	7.69	0.0	0.0	13.65 [4.18]	0.0 [0.0]	0.1 [0.53]	14.17 [3.62]	0.0 [0.0]	0.0 [0.0]
6	7.92	0.0	0.0	10.73 [2.09]	0.0 [0.0]	0.0 [0.01]	11.24 [4.7]	0.0 [0.0]	0.0 [0.0]
7	7.27	0.0	0.0	11.86 [2.93]	0.1 [0.54]	0.0 [0.0]	12.48 [3.5]	0.0 [0.0]	0.0 [0.0]
8	9.55	0.0	0.0	13.48 [3.52]	0.34 [1.47]	0.0 [0.0]	11.97 [2.97]	0.26 [0.97]	0.0 [0.0]
9	0.0	35.59	0.0	0.02 [0.13]	32.55 [9.13]	0.0 [0.01]	0.03 [0.11]	31.51 [10.48]	0.0 [0.0]
10	0.0	20.38	0.0	0.9 [1.63]	21.8 [8.44]	0.0 [0.01]	1.44 [2.69]	20.51 [8.5]	0.0 [0.0]
11	0.0	18.67	0.0	0.02 [0.08]	23.79 [5.62]	0.29 [1.29]	0.17 [0.56]	24.68 [5.55]	0.0 [0.0]
12	0.0	17.17	0.0	9.58 [1.95]	0.1 [0.32]	0.07 [0.25]	9.36 [1.96]	0.02 [0.09]	0.03 [0.09]

not expect planned paths to exactly match the demonstrations. Two notable types of error resulted in sub-optimal plans, namely cases where planned paths clip the edge of higher-sensitivity objects, and cases where planned paths simply result in higher cost than the demonstration. In both cases, errors are indicated by high standard deviations; this is expected if a small number of the planned paths exhibit particularly sub-optimal behavior. These errors are caused by the limited time available to RRT*, which restricts the number of goal states sampled and the refinement of the path. Results for 60-min planning times shown in Table 1 show that in most cases, increased planning time reduces these errors. Note that the high planning times used here are partially a consequence of our puncture test, which adds considerable computation in addition to the deformation cost function.

5.1.2 Generalization of Recovered Parameters

The importance of recovering sensitivity parameters is not to reproduce the demonstrations, since these could simply be replayed; rather, recovering the sensitivity parameters allows us to generalize the behavior displayed in the demonstrations to other tasks in the test environment. To demonstrate that the recovered sensitivity parameters generalize, we performed a set of tests shown in Fig. 6. Starting from one of the demonstrations (demonstration task 6), we adjusted the target point and inserted rigid obstacles that block the demonstrated path. As shown in Fig. 6, our planner produces paths that exhibit the same behavior as the demonstration path; while the new path differs from the demonstration and thus results in different cost, the preferential deformation of the blue object over the green one indicates that the expert's preference was correctly captured.

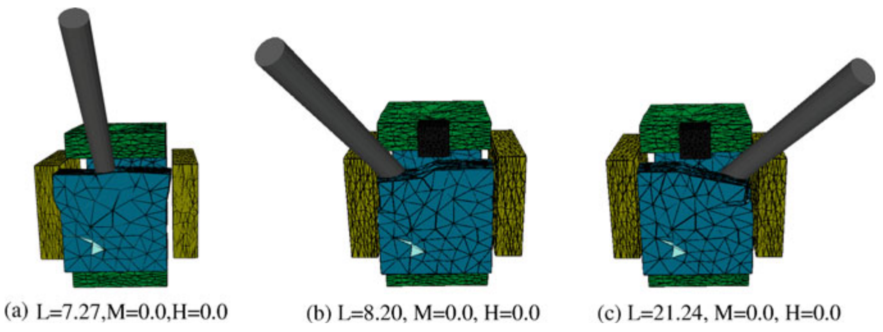


Fig. 6 Paths planned to show the generality of recovered sensitivity values, **a** goal configuration of demonstration 6, and **b**, **c** two goals of paths planned with target points offset from the center of the environment when the direct path from start to target is blocked by a rigid obstacle (*black*)

5.2 Automatic Generation of Demonstration Tasks

Using the same test environment, we tested our active learning method for automatically collecting demonstration tasks. Examples of these demonstration tasks are shown in Fig. 2. Unlike the conservative approach discussed previously, which used demonstrations between all pairs of adjacent objects, the active learning method generates only enough tasks to form a ranking of all objects in the environment. We tested the active learning method in the same test environment as above and allowed it to select a subset of tests from the set of comprehensive demonstrations. Using this method, between 8 and 10 demonstrations were required to capture features for all objects, compared to the 12 used by the conservative approach. As before, 100 sample paths were generated around each demonstration, and sensitivities were recovered using the PIIRL optimization problem. Since the active learning process involves some random selections, we ran 15 trials; 10 demonstrations were required in 14 cases, and 8 demonstrations in 1 case, with average recovered sensitivities (average [std.dev.]) being $L_1 = 0.100[0.0]$, $L_2 = 0.101[0.0002]$, $M_1 = 2.858[0.012]$, $M_2 = 8.630[0.071]$, $H_1 = 984.12[29.272]$, $H_2 = 999.509[0.165]$. Comparing these results with the sensitivities learned using the full set of demonstrations (see Sect. 5.1), we observe that the values are not meaningfully different, which shows that the active learning method can infer very similar sensitivity relationships with fewer demonstrations.

5.3 Physical Environment Tests

In addition to testing with our simulated environment, we have also applied our framework to a planar physical test environment shown in Fig. 7 with an L-shaped block, similar to those used in our previous work [16]. Like our previous work, the use of a planar 3DoF environment allows for the deformation of objects in the environment to be tracked in real time by an overhead camera. Paths in the environment were planned using the same RRT* planner as before, albeit in $SE(2)$.

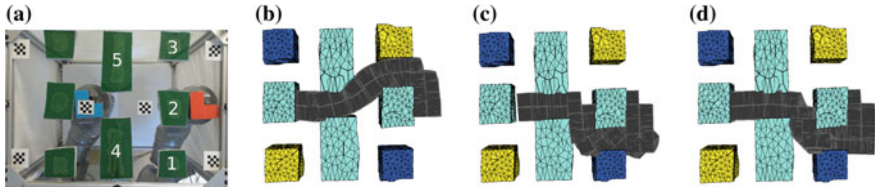


Fig. 7 Testing for our physical test environment (**a**), with objects numbered and start (*red*) and goal (*blue*) states shown. Swept volumes of **b** path planned with uniform object sensitivity values, **c** demonstration path, and **d** path planned with recovered sensitivity values

Table 2 Deformation comparison for the five right-hand objects in our physical test environment between a path planned with uniform object sensitivity values, the demonstrated path, and a path planned using the recovered sensitivity values. Reported deformation values are in pixels

	Object deformation				
	O_1	O_2	O_3	O_4	O_5
Uniform	0	3367	2442	0	554148
Demonstration	23451	0	0	0	35222
Recovered	51569	38798	0	0	73013

For comparison purposes, we first planned using uniform sensitivity values for all objects, as shown in Fig. 7b. A demonstration path through a narrower, higher-deformation passage was provided using our demonstration capture environment, as shown in Fig. 7c. As with the simulated environment, 100 samples were generated around the demonstration, and object sensitivity values $O_1 = 1.00$, $O_2 = 200.00$, $O_3 = 100.03$, $O_4 = 200.00$, $O_5 = 36.21$ were recovered using a lower bound of 1, upper bound of 200, and initial value of 100. These parameters are expected, as the demonstration path deforms O_1, O_4 and O_5, while avoiding the other objects. Planning using the recovered values is shown in Fig. 7d; planning was performed with a planning time of 5 min. Following planning, all three paths were executed in our test environment by an industrial robot, with object deformations tracked by our tracking camera and reported in Table 2. As before, we do not expect the planned path to exactly match the demonstration; in particular due to the narrow low-cost passages in the environment, it is unsurprising that the planned path has significantly higher cost than the expert demonstration. However, the planned path does avoid O_3, instead preferring the passage between O_1 and O_2, which matches the preferences demonstrated by the expert.

6 Conclusion

We have developed a framework for recovering sensitivities of deformable objects so that our motion planners imitate the behavior of expert users in deformable environments. By formulating the problem of motion planning in deformable environments in terms of generating optimal paths that minimize deformation, we can recover object sensitivity parameters from demonstrated optimal paths using IOC. We also propose an active learning algorithm to generate demonstration tasks. Our framework has two advantages over existing similar techniques. First, by using sampling-based techniques for IOC that avoid the need to solve the forward problem and sampling-based asymptotically-optimal planners, our framework is more applicable to higher-dimensional problems than existing approaches. Second, our method for automatically generating demonstration tasks for users to perform reduces the number of demonstration tasks needed to capture the desired behavior. We tested our framework in simulated and physical test environments, and showed that it recovers object sensitivities suitable for planning paths that imitate the behavior of expert demonstrations. We also showed that these preferences can generalize to new tasks.

References

1. Abbeel, P., Ng, A.Y.: Apprenticeship learning via inverse reinforcement learning. In: ICML (2004)
2. Alterovitz, R., Goldberg, K.: Motion Planning in Medicine: Optimization and Simulation Algorithms for Image-Guided Procedures. Springer Tracts in Advanced Robotics. Springer, Berlin (2008)
3. Chen, L., Rong, Y.: Linear time recognition algorithms for topological invariants in 3D. In: International Conference on Pattern Recognition (2008)
4. Coumans, E.: Bullet 2.73 Physics SDK Manual (2010)
5. Faure, F., Gilles, B., Bousquet, G., Pai, D.K.: Sparse meshless models of complex deformable solids. In: ACM Transactions on Graphics pp. 73–73 (2011)
6. Gibson, S.F.F., Mirtich, B.: A survey of deformable modeling in computer graphics. Technical report, Mitsubishi Electric Research Laboratories (1997)
7. Irving, G., Teran, J., Fedkiw, R.: Invertible finite elements for robust simulation of large deformation. In: ACM SIGGRAPH/Eurographics Symposium on Computer Animation, SCA '04. New York, New York, USA (2004)
8. Jiménez, P.: Survey on model-based manipulation planning of deformable objects. Robot. Comput.-Integr. Manuf. **28**(2), 154–163 (2012)
9. Kalakrishnan, M., Chitta, S., Theodorou, E., Pastor, P., Schaal, S.: STOMP: stochastic trajectory optimization for motion planning. In: ICRA (2011)
10. Kalakrishnan, M., Pastor, P., Righetti, L., Schaal, S.: Learning objective functions for manipulation. In: ICRA (2013)
11. Kalman, R.: When is a linear control system optimal? J. Basic Eng. (1964)
12. Karaman, S., Frazzoli, E.: Sampling-based algorithms for optimal motion planning. Int. J. Robot. Res. **30**(7), 20 (2010)
13. Maris, B., Botturi, D., Fiorini, P.: Trajectory planning with task constraints in densely filled environments. In: IROS (2010)
14. Müller, M., Dorsey, J., McMillan, L., Jagnow, R., Cutler, B.: Stable real-time deformations. In: ACM SIGGRAPH/Eurographics Symposium on Computer Animation, SCA '02. New York, New York, USA (2002)
15. Ng, A.Y., Russell, S.J., et al.: Algorithms for inverse reinforcement learning. In: ICML (2000)
16. Phillips-Grafflin, C., Berenson, D.: A representation of deformable objects for motion planning with no physical simulation. In: ICRA (2014)
17. Ratliff, N., Silver, D., Bagnell, J.A.D.: Learning to search: functional gradient techniques for imitation learning. Auton. Robots (2009)
18. Şucan, I.A., Moll, M., Kavraki, L.E.: The open motion planning library. IEEE Robot. Autom. Mag. **19**(4), 72–82 (2012). http://ompl.kavrakilab.org
19. Ziebart, B.D., et al.: Maximum Entropy Inverse Reinforcement Learning. In: AAAI, pp. 1433–1438 (2008)

Computer-Aided Compositional Design and Verification for Modular Robots

Tarik Tosun, Gangyuan Jing, Hadas Kress-Gazit and Mark Yim

1 Introduction

Modular reconfigurable robot systems have been studied extensively for several decades. These systems distinguish themselves from conventional robotic systems in their ability to transform into different shapes to address a wide variety of tasks. They promise to be versatile, robust, and low cost [27]. Dozens of groups have built different kinds of reconfigurable robots [6, 11], and introduced approaches for programming them [18, 21, 30]. Over 800 papers, a book [9], and a survey [28] have been written on the subject.

This versatility places an additional burden on the user, because solving problems with modular robots involves not only writing programs, but also determining the best physical form for the task at hand. If this complexity is not appropriately managed, it will present a significant barrier to using modular robots to address practical tasks [26]. If the user is free to create any new design to solve a new task, but must program the design from scratch every time, creating new designs will be a huge amount of effort, and the advantage of versatile modular hardware will be defeated.

Tarik Tosun and Gangyuan Jing contributed equally to this work.
This work was funded by NSF grant numbers CNS-1329620 and CNS-1329692.

T. Tosun (✉) · M. Yim
University of Pennsylvania, Philadelphia, PA, USA
e-mail: tarikt@grasp.upenn.edu

M. Yim
e-mail: yim@grasp.upenn.edu

G. Jing · H. Kress-Gazit
Cornell University, Ithaca, NY, USA
e-mail: gj56@cornell.edu

H. Kress-Gazit
e-mail: hadaskg@cornell.edu

A. Bicchi and W. Burgard (eds.), *Robotics Research*, Springer Proceedings in Advanced Robotics 2, DOI 10.1007/978-3-319-51532-8_15

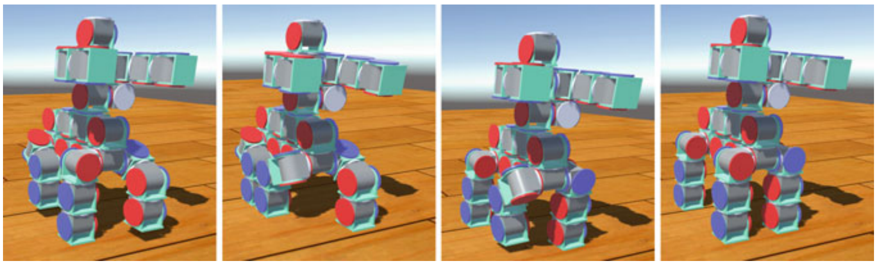


Fig. 1 The Centaur is a mobile manipulator made of 29 modules. The framework we present provides tools that help users quickly create, program, and verify complex designs like the Centaur by composing existing designs and behaviors from a library

Software modularity is a well-established practice for developing large maintainable systems and avoiding duplication of effort. In robotics, software behaviors are inextricably linked to the hardware they control, resulting in challenges to making modularity effective. Significant progress has been made on these fronts in traditional robotics, most notably ROS [17] which provides inter-process communication and standard libraries for common robot tasks, as well as verification tools [8]. In modular robotics, the challenge is different. Modular robot systems are not usually optimized for specific tasks, so in order to use them most effectively, we must take advantage of their flexibility. To do so, a user must be able to generate and verify configurations and behaviors as quickly as possible.

Toward that end, we present a design framework that facilitates the rapid creation of new configurations and behaviors through composition, and tools to verify them while they are being created. New configurations are created by combining existing sub-configurations, for example combining a four-legged walking robot with a two-fingered gripper to form a mobile manipulator, like the "Centaur" configuration shown in Fig. 1. Behaviors are associated with each configuration, so that when sub-configurations are composed, their associated behaviors are immediately available for use. The Centaur in Fig. 1, for example, can immediately execute the walking behavior of its component four-legged base. We introduce a new motion description language (Series-Parallel Action Graphs, Sect. 4.2) that facilitates the rapid creation of complex behaviors by composition of simpler behaviors (for example, composing "Grasp" and "Walk" behaviors to make the Centaur pick up and carry an object). We provide tools that automatically verify configurations and behaviors during the design process, identifying conflicting commands, self-collision, loss of gravitational stability, and forces exceeding the limits of safety for actuators and connectors. This allows users to identify problems early and iterate quickly on complex new designs. In addition to verification, users can evaluate their configurations and behaviors in a physics-based simulator. The software we have developed is open-source, and will be made freely available online at: http://modlabupenn.org/compositional-design/.

The remainder of this paper provides a description of the structure and algorithmic components of our framework. In Sect. 2, we discuss relevant background material. In

Sect. 3 we introduce terminology and concepts used elsewhere in the paper. In Sect. 4, we describe the algorithmic basis for the three major components of our framework - design composition, behavior composition, and verification. In Sect. 5, we discuss the open-source software tools used to implement our framework. In Sect. 6, we provide examples highlighting important aspects of the framework, including a demonstration of the user's workflow.

2 Related Work

In some respects, our work parallels the efforts of Mehta [14] and Bezzo [1], who aim to create and program printable robots from novice users' design specifications. Users create new designs by composing existing elements from a design library, and appropriate circuitry and control software are automatically generated as physical designs are assembled. The framework we present is intended specifically for modular robots, and consequently the workflow and design considerations are fundamentally different from that presented by Mehta and Bezzo. In traditional robot design (or printable robot design), hardware and software are somewhat decoupled - hardware is designed and built once, and then programmed many times. In the case of a modular robot system, the system can be reconfigured to meet new tasks, so hardware configuration and behavior programming go hand in hand. We intend our system to be fast enough that the user could conceivably develop and program a new configuration for every new task - configurations are built once, and programmed once. Where Mehta et al. provide many facilities to generate and verify low-level behaviors (e.g. motor drivers appropriate for motors), we do so for high-level behaviors.

A significant amount of work has been done in developing behaviors and software for modular robots. Genetic algorithms have been applied for the automated generation of designs and behaviors [7]. Other work has focused on distributed control [24], hormone-based control [18], and central pattern generators [20].

Efforts have also been made to generate behaviors by automatically identifying the "role" a module should play based on its place in a connected structure [21]. Functionality is propagated downward: based on a high-level goal (like "walk") and a connected structure of modules, functional sub-structures (like legs and a spine) are automatically identified, and modules are directed to execute appropriate roles in a distributed fashion. In our work, modular structures are similarly associated with appropriate behaviors. Rather than identifying roles in a top-down fashion, we build designs with the desired functionality from the bottom up. The user creates new designs by composing sub-components and associated behaviors from a library, building a new structure that can definitely execute the desired behavior.

While significant progress has been made in the automated generation of modular robot behaviors, automated systems are not yet capable of making modular robots truly useful in practice [28]. The need for new programming techniques to manage the complexity of modular robot systems has been acknowledged in the literature

[26]. Historically, gait tables have been a commonly used format in which open-loop kinematic behaviors can be easily encoded [25]. Phased automata have also been presented as a way to easily create scalable gaits for large numbers of modular robots [30]. In this paper, we introduce a novel motion description language that enables users to quickly create behaviors for modular robots.

A number of robot simulators have been developed, including simulators specifically for modular robots [3]. For our work, we opted to use Gazebo [10] because of its growing popularity in the robotics community. While our software currently only supports the SMORES robot [5], other modular robot designs can easily be incorporated. Future work includes incorporating support for the CKBot robot [4].

Our framework assists users in verifying design validity by identifying conflicting commands, self-collision, loss of gravitational stability, and forces exceeding the limits of safety for actuators and connectors. In existing literature, some of these conditions have been checked in the context of modular robot reconfiguration planning [2] and motion planning [29]. To our knowledge, there is no modular robot design tool that verifies these conditions to provide assistance to a human designer.

3 Definitions

In this section, we present concepts and terms which will be used later in the paper.

Definition 1 (*Module*) A module is a small robot that can move, respond to commands, and attach to other modules. Formally, we define a module as $\mathscr{M} = ({}^{\mathscr{W}}D^{\mathscr{M}}, X, A, K)$. The rigid body *displacement*, ${}^{\mathscr{W}}D^{\mathscr{M}} \in SE(3)$ gives the position and orientation of the module body frame in the world frame $\mathscr{W}$. The *state* of the module, $X = [x_1, x_2, \ldots, x_d]$, is a d-dimensional vector representing the d joint angles of the module. $A = \{a_1, a_2, \ldots, a_k\}$ is the set of *attachment points* where the module can connect to other modules. The module's *forward kinematics function*, $K : (X, a_i) \rightarrow SE(3)$ returns ${}^{\mathscr{M}}D^{a_i}$ (the displacement of attachment point a_i in the module frame) as a function of X. Figure 2 shows a schematic representation of a module with four attachment points.

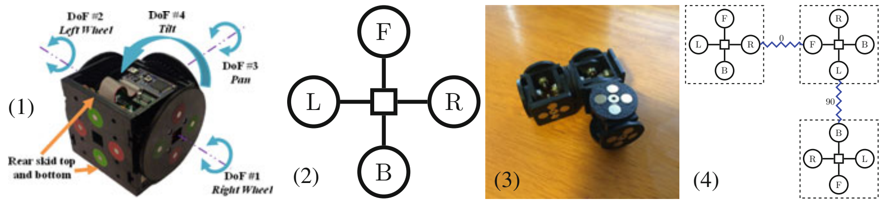


Fig. 2 From *left*: **a** A photo of a SMORES module with four attachment points (*left*, *right*, *front*, and *back*), **b** its graphical representation, **c** a photo of a configuration with three modules, and **d** its graphical representation

In this paper, we demonstrate our framework using a homogeneous modular robot system (all modules are identical). The framework could be extended to heterogeneous systems by including more information in the definition of a module - for example, if the system used multiple kinds of connectors, labels on the attachment points could be included.

Definition 2 (*Configuration*) A *configuration* is a contiguous set of connected modules which we treat as a single robot. The identity of a configuration is determined by its connective structure; configurations can be represented by graphs with nodes representing modules and edges representing connections between modules. Individual modules are considered interchangeable (as long as they are of the same type).

In this paper, we present an object-oriented design framework for modular robot systems, and treat configurations as the fundamental objects. Rather than defining configurations only by the topology of their component modules, we define them recursively, as being composed of connected sub-configurations. A single module is considered the smallest configuration.

Formally, we define a configuration as $\mathscr{C} = (C_, \gamma, M, E, \delta, X, B)$. Here, $C = \{\mathscr{C}_1, \mathscr{C}_2, \ldots, \mathscr{C}_q\}$ is a set of sub-configurations. $\gamma : C \to 2^M$ is a function mapping a configuration $\mathscr{C}_i \in C$ to its set of modules, $M = \bigcup_{\mathscr{C} \in C} \gamma(\mathscr{C})$. E is a set of connections between modules. Elements of E are pairs of attachment points, $(\mathscr{M}_i.a_i, \mathscr{M}_j.a_j) \in E$, where $\mathscr{M}_i, \mathscr{M}_j \in M$, $\mathscr{M}_i \neq \mathscr{M}_j$, and $a_i \in \mathscr{M}_i.A, a_j \in \mathscr{M}_j.A$. The orientation of one attachment point relative to another is represented by the labeling function $\delta : E \to SO(3)$, returning ${}^{\mathscr{M}_i.a_i}R^{\mathscr{M}_j.a_j}$. The *state* of the configuration is $X = \bigcup_{\mathscr{M}_i \in M} \mathscr{M}_i.X$. Finally, associated with each configuration is a set of *behaviors* B (see Definition 3).

Figure 2 shows a photo of a configuration composed of three modules, each with four attachment points, and its graphical representation. Blue zigzag lines represent connections between modules, and the label of each connection shows the angle offset of that connection. We can compute forward kinematics for the entire configuration by composing displacements module-to-module. Let any module $\mathscr{M}_f \in M$ have fixed displacement ${}^{\mathscr{W}}D^{\mathscr{M}_f}$ in the world frame. Let $\mathscr{M}_i : (\mathscr{M}_i.a_i,\ \mathscr{M}_f.a_f) \in E$ be connected to $\mathscr{M}_f$. We can find ${}^{\mathscr{W}}D^{\mathscr{M}_i}$ by composing displacements as follows:

$$\begin{aligned} {}^{\mathscr{W}}D^{\mathscr{M}_i} &= [{}^{\mathscr{W}}D^{\mathscr{M}_f}][{}^{\mathscr{M}_f}D^{a_f}][{}^{a_f}D^{a_i}][{}^{\mathscr{M}_i}D^{a_i}]^T \\ &= [{}^{\mathscr{W}}D^{\mathscr{M}_f}][K_f(X_f, a_f)] \begin{bmatrix} \delta(e) & 0 \\ 0 & 1 \end{bmatrix} [K_i(X_i, a_i)]^T \end{aligned}$$

where $e = (\mathscr{M}_i.a_i,\ \mathscr{M}_f.a_f)$. To find the world-frame displacements of all other modules, we may traverse the connections of the configuration, repeatedly composing displacements in the manner above.

Definition 3 (*Behavior*) A *behavior* $B : (t, X) \to X_{set}$ is a programmed sequence of movements defined over the joints of a specific configuration, and intended to produce a desired effect - a gait for walking is one example. Behaviors determine the controller setpoints X_{set} for a configuration as a function of state X and time t.

In this paper, we represent behaviors as series-parallel action graphs, described in detail in Sect. 4.2.

Definition 4 (*Controller*) A *controller* is a position or velocity servo for one DoF of a modular robot. A controller takes as input a desired position or angular velocity, and drives the error between the desired and actual state of the DoF it controls to zero over time.

4 Approach and Algorithm

The three major components of our framework are configuration composition, behavior composition, and verification of configurations and behaviors. Together, these three components provide a streamlined workflow to quickly create functional robots by leveraging an existing library of designs and behaviors. Combining existing designs and behaviors into new ones allows users to create large, complicated, functional designs.

4.1 Configuration Composition

Before discussing configuration composition, we will first define a set of connections E_C between configurations in a given set C as $(\mathscr{C}_i.\mathscr{M}_i.a_i, \mathscr{C}_j.\mathscr{M}_j.a_j) \in E_C$, where $\mathscr{C}_i, \mathscr{C}_j \in C, \mathscr{M}_i \in \gamma(\mathscr{C}_i), \mathscr{M}_j \in \gamma(\mathscr{C}_j)$, and $a_i \in \mathscr{M}_i.A, a_j \in \mathscr{M}_j.A$. We form a graph with configurations in C as nodes and connections in E_C as edges.

Given a set of configurations C and a set E_C of connections between them, configuration composition combines all configurations in C to a single configuration $\mathscr{C}^*$ that includes all modules and connections from C and E_C. The composed configuration is $\mathscr{C}^* = (C^*, \gamma, M, E, \delta, X, B)$, where C^*, M, E, and B are the unions of the corresponding sets of the sub-configurations in C.

4.2 Behavior Composition: Series-Parallel Action Graphs

The modular robotics community has developed a number of methods to create behaviors, including gait tables [25], phased automata [30], hormone-based control [18], and role-based control [21]. Phased automata, hormone, and role-based control are typically used to specify a single, cyclic behavior (such as a gait for locomotion) in a distributed fashion. These methods have good scaling and robustness properties, but are not well-suited to specifying the non-cyclic, globally coordinated behaviors required for many tasks (like picking up and moving an object with an arm).

The simplicity and clarity of gait tables makes them appealing for our application. However, gait tables are often difficult to compose or re-use, and also hard to scale to very complicated designs. The motion description language we present allows low-level behaviors to be combined in series and parallel to create new higher-level behaviors, encapsulating complexity and facilitating code re-use. The resulting programs are expressive, and have a nested structure that is easy to understand and debug.

The atoms of the language are called *actions*. Similar to a single entry of a gait table, an action specifies a controller setpoint for a single DoF of a module. Unlike a gait table entry, actions do not have explicit timestamps. Rather, each action has an associated interrupt condition, which is a boolean function of the (sensed) state of the robot. Similar to a state transition in a finite state machine (FSM), when the interrupt condition is met the action is considered complete, and execution moves on to the next action. Interrupts allow the programmer to precisely specify behaviors in a natural way: rather than specifying a timed sequence of motions, the programmer specifies an ordered sequence of actions and has some assurance that an action will not begin until the robot has actually achieved the goal state of the previous action. Actions may optionally include a timeout, which causes the action to be considered complete automatically once time runs out.

An important distinction between actions in our language and states in a traditional FSM is that multiple actions may execute in parallel. Actions are combined through parallel and series composition to create behaviors. When two actions are composed in series, the second begins when the first ends. When composed in parallel they begin simultaneously, and the following actions do not begin until both complete. A behavior created using these operations is a directed acyclic graph of actions with series-parallel structure [23]; Fig. 3 provides a visual example.

As an example, consider the car design shown in Fig. 4. To create a low-level "drive-forward" behavior, we simply command all of its wheels to spin in parallel. The car steers by swiveling its central steering column, so a "turn right" behavior can be similarly achieved by commanding parallel actions for the steering column joints. With these low-level behaviors established, we can command trajectories through series composition. For example, if we name our car configuration `c`: `c.square = **series** (c.drive, c.turn, c.drive, c.turn, c.drive, c.turn, c.drive, c.turn)`.

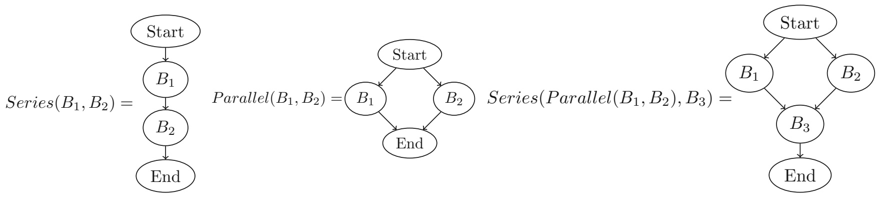


Fig. 3 Series and parallel composition

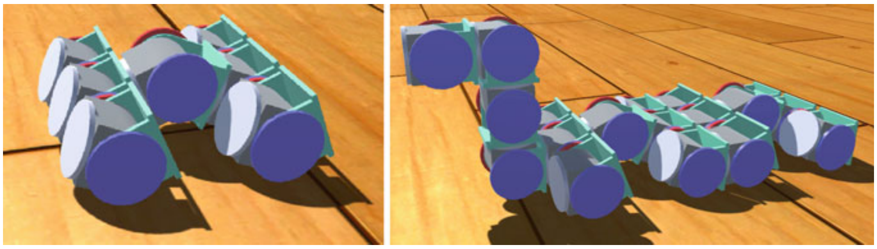


Fig. 4 Car design (*left*) and backhoe design (*right*)

This paradigm allows low-level behaviors to be coded quickly and easily. However, the real value comes from its ability to combine behaviors in layers and quickly generate behaviors for complicated designs. This works particularly well when designs are made by composing smaller designs. For example, we can develop "drive" and "turn" behaviors for the 18-module backhoe shown in Fig. 4 (name `bh`) by composing behaviors of its component car designs (named `c1` and `c2`): `bh.drive =` **`parallel`** `(c1.drive, c2.drive)`, and `bh.turn =` **`parallel`** `(c1.turn, c2.turn)`. Each of these one-line statements commands the movement of 28 degrees of freedom.

We can extend this further to generate high-level behaviors for the backhoe. Suppose that the arm has a laser rangefinder attached to the end, and that we've already created a "scan" behavior that sweeps or rotates the sensor. We might create a "patrol" behavior that scans continuously while driving in a square: `bh.patrol =` **`parallel`** `(bh.scan, bh.square)`. Or, if we only want the robot to scan the corners of a room, we can precisely specify this using lower-level behaviors: `bh.cornerScan =` **`series`** `(bh.drive, bh.scan, bh.turn, bh.drive, bh.scan, bh.turn, ...)`.

Building behaviors in this layered fashion makes it easy to re-use code and quickly generate complicated behaviors. Of course, there is no guarantee that two composed behaviors will be compatible; it is possible to mistakenly create behaviors that are impossible or dangerous to execute. For this reason, we provide verification tools that automatically identify problems - for example, if two behaviors composed in parallel commanded the same DoF simultaneously (a problem we call *behavior conflict*), this would be automatically identified. Our verification tools are explained in detail in Sect. 4.3.

Our emphasis on abstraction begs the question: why not use a a more fully-featured plan execution model such as behavior trees [13], parallel-hierarchical finite state machines [19] or even a traditional object-oriented programming language (like Java or C++)? Our decision was driven by the tradeoff between complexity and ease-of-use: given our desire for simplicity and speed of programming, we chose a minimal paradigm with only two composition operations. The language is quite expressive: we have used it to develop complex behaviors for large designs (see Sect. 6). The language is also limited: it does not yet include conditional statements, iteration, or access to environmental sensor information (other than joint angles). In the future, we hope to include these capabilities without sacrificing ease-of-use.

4.3 Verification of Configurations and Behaviors

Verification of Configurations: In Sect. 4.1, we introduced the definition of configuration composition. During the design process, a user might attempt to compose configurations in a way that is unstable or physically impossible. By incorporating existing algorithms into the design process, we provide tools to automatically verify designs during construction, saving time that would otherwise be spent simulating or testing invalid designs.

Given a configuration $\mathscr{C}$ and state X_0, we consider $\mathscr{C}$ to be valid in state X_0 if it is gravitationally stable and free from self collision between modules. A robot is gravitationally stable when it is balanced, and gravity does not create any net moment on it. If this condition is not met, the robot could tip over and suffer damage. A *self-collision* occurs when two different parts of the configuration are commanded to occupy the same location in space. Self-collisions can also cause damage, and are almost always unwanted.

To determine gravitational instability and self-collision, we assume that the geometric, kinematic, and mass information for each module are available. To check for self-collision, the positions and orientation of all modules are obtained through forward kinematics as in Definition 2. Our tool checks self-collision by approximating modules as spheres, and checking the distance (radius) between all pairs. Due to this approximation, false-positive collisions might be detected. When this happens, a user can easily spot the faulty detection in the final configuration and choose to ignore such warning. More sophisticated techniques are available which efficiently produce exact results [15], at the cost of higher complexity. To offer instant feedback to the user when designing the configurations, we check gravitational stability by computing the location of the center of mass of the configuration based on the known kinematics and mass properties of the modules. We find the set of modules that have minimal position in the z direction and consider them to be in contact with the ground plane, treating their centroids as an approximate set of ground contact points. If the projection of the configuration center of mass onto the ground plane lies within the convex hull of the ground contact points, gravity exerts no moment and the configuration is stable.

Verification of Behaviors: In Sect. 4.2, we introduced a novel motion description language for modular robots. Like configurations, behaviors are automatically verified as the user composes them. In addition to being free from self-collision and gravitationally stable during execution, a valid behavior also must not exceed the actuator or connector force limits of the modules, and must be free from behavior conflict.

To verify a behavior with time duration T_B, we discretize execution with a preset sampling time t_B. At each time step, we first detect behavior conflict by checking if different commands are given to the same joint of a module simultaneously. If there is no behavior conflict, we update the positions and orientations of all modules in the configuration based on the commands. We then check for self-collision and gravitational instability, using the methods described above. A behavior that results in self-collision during a single time step is considered invalid. For gravitational stability, we specify a time bound $t_{max} > t_B$. A behavior is considered unstable if it includes any period of instability longer than t_{max}, or if the behavior is unstable at time T_B (at the end).

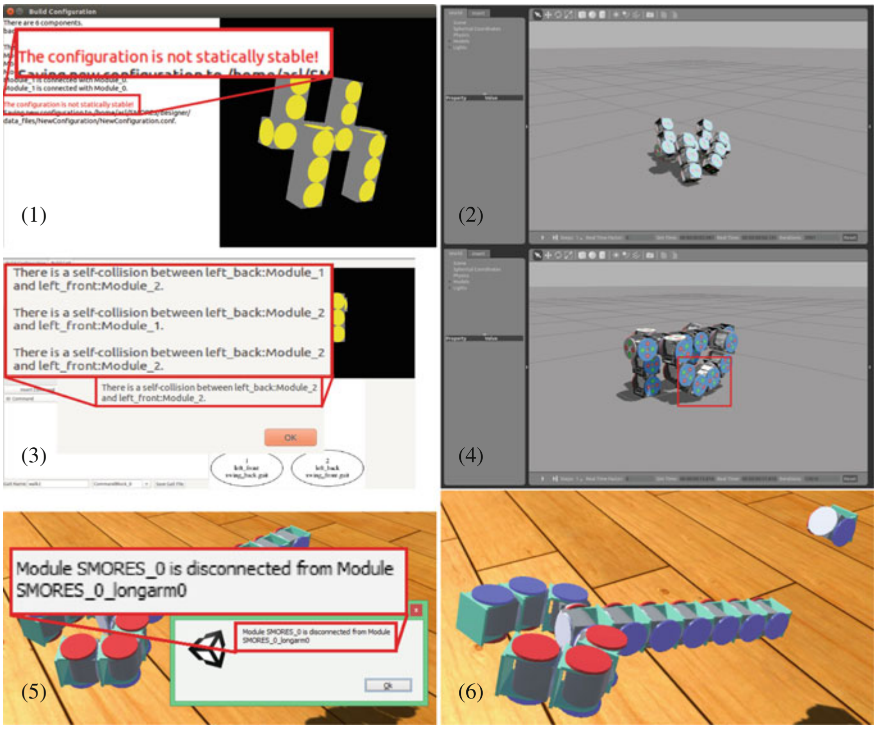
To check force limits, unlike other verifications for behaviors, we use an existing physics engine to detect unsafe conditions during simulation. By setting the maximum force that can be supported by connectors and exerted by joints, we are able to identify unsafe behaviors if we detect, during the behavior execution, any undesired module disconnection or a mismatch between any joints target position and actual position.

The need for verification becomes more important as design complexity increases. Consider a four-legged Walkbot example shown in Fig. 5a. If the user sets two of the connections with different angle offset, the composed Walkbot configuration will have two legs pointing in the opposite direction of the other two legs, as shown in Fig. 5a. Since the projection of the configuration's center of mass now falls out of the supporting base, the program will warn the user that the configuration is not gravitationally stable. As shown in Fig. 5b, in simulation the configuration quickly fell to the ground due to the instability as warned by the program.

Verification of behaviors also aids the user in creating valid and safe robot behaviors. When designing the walking behavior for the Walkbot, if the user commands the front and rear leg at the same side of the robot to swing toward each other at the same time, the program will warn the user that there will be collision between modules in this behavior, as shown in Fig. 5c. The image shown in Fig. 5d demonstrates the moment of collision during simulation.

There is a trade-off between the correctness and the efficiency of the verification. By reducing the sampling time t_b, more potential self-collisions or gravitational instability can be detected with the cost of longer computation time. However, since there is no real-time requirement (verification is done during the design process, not at runtime), the computational cost of fine-resolution verification is worthwhile in most cases.

Modules all have limits on the maximum force that is available to maintain connections with other modules and to drive each joint to desired positions. Thus, it is crucial to notify the user if there is no sufficient force from the module's hardware to execute a behavior while maintaining all module connections. As shown in Fig. 5e, the program detected a disconnection when the user tried to lift a long cantilever arm. Figure 5f demonstrates the disconnection in simulation.



There are 6 components.

The configuration is not statically stable!

Module_1 is connected with Module_0.
Module_1 is connected with Module_0.

The configuration is not statically stable!
Saving new configuration to /home/asl/[illegible]/designer/
data_files/NewConfiguration/NewConfiguration.conf.

There is a self-collision between left_back:Module_1 and left_front:Module_2.

There is a self-collision between left_back:Module_2 and left_front:Module_1.

There is a self-collision between left_back:Module_2 and left_front:Module_2.

Module SMORES_0 is disconnected from Module SMORES_0_longarm0

Fig. 5 From *top* and *left*: **a** The design tool warns the composed configuration is not gravitationally stable; **b** The robot fell to ground plane due to instability, in simulation; **c** The design tool indicates there is collision during the behavior execution; **d** Two feet of the robot collided during simulation; **e** The design tool indicates there is undesired disconnection; **f** The configuration disconnected during simulation

5 Implementation

Our implementation currently supports only the SMORES modular robot [5], but could easily incorporate any other modular robot for which kinematic, geometric, and mass information is available. Each SMORES modules has four DoF - three continuously rotating faces called *turntables* and one central hinge with a 180° range of motion (Fig. 2a). The DoF marked 1, 2, and 4 have rotational axes that are parallel and coincident. Each SMORES module can drive around as a two-wheel differential drive robot. SMORES modules may connect to one another via magnets on each of their four faces, and are capable of self-reconfiguration. Formally, we denote the state of a SMORES module as $X = \{\theta_L, \theta_R, \theta_F, \theta_B\}$ and the set of attachment points as $A = \{L, R, T, B\}$.

A design interface was implemented to aid users in building complex configurations and behaviors from a set of basic configurations and associated behaviors, and then verifying their correct execution. We separated the design tool into two main

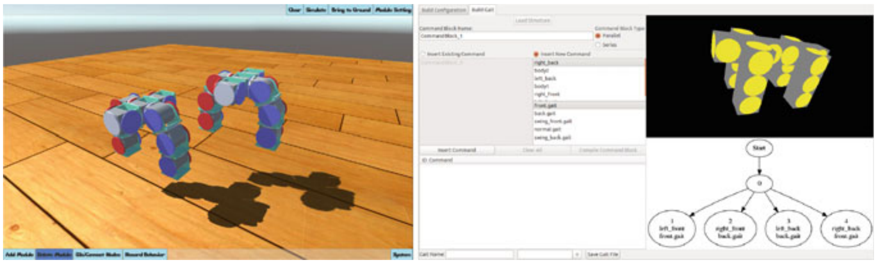


Fig. 6 GUIs for configuration builder (*left*) and behavior builder (*right*)

parts: a configuration builder and a behavior builder. Given a set of basic configurations (which could be just single modules), the configuration builder allows users to combine basic configurations by choosing connection nodes on each configuration, as demonstrated in Fig. 6. In addition, the configuration builder warns users when the composed configuration is not stable or contains self-collisions. It does so without the computation complexity of a physical simulator, e.g. Gazebo [10].

Given a configuration, the behavior builder aids users in designing behaviors by composing existing behaviors in series and parallel. Figure 6 illustrates a new behavior composed by putting four basic behaviors in parallel. Similar to the configuration builder, the behavior builder warns users if there are self-collisions or behavior conflicts during the execution of composed behaviors, without the need of a physics-based simulator. To check force limits for connections and actuators, we create a model of the module in the PhysX [16] physics engine with Unity3D [22], and specify joint and connection force limits.

6 Examples and User Perspective

Our eventual intention is to develop a large library of configurations and associated behaviors which are available to all users of our framework, analogous to the standard libraries of major programming languages. The compositional nature of our framework will allow users to rely heavily on the library when approaching new tasks, allowing them to create sophisticated robots very quickly.

As a first step toward a standard library, we present a small library of configurations in Fig. 7. Configurations in the library are organized by *order*, defined recursively as follows: a single module is an order-zero configuration, and the order of all other configurations is one greater than the largest order of the sub-configurations from which it is composed. Each configuration has an associated set of behaviors, which the user can compose to accomplish tasks. New behaviors for a higher-order configuration can be created by composing the behaviors of its component sub-configurations.

For the library to be most effective, the set of configurations and behaviors available at each level (and especially at the lowest levels) should provide a rich set of functionalities without presenting the user with an overwhelming number of options.

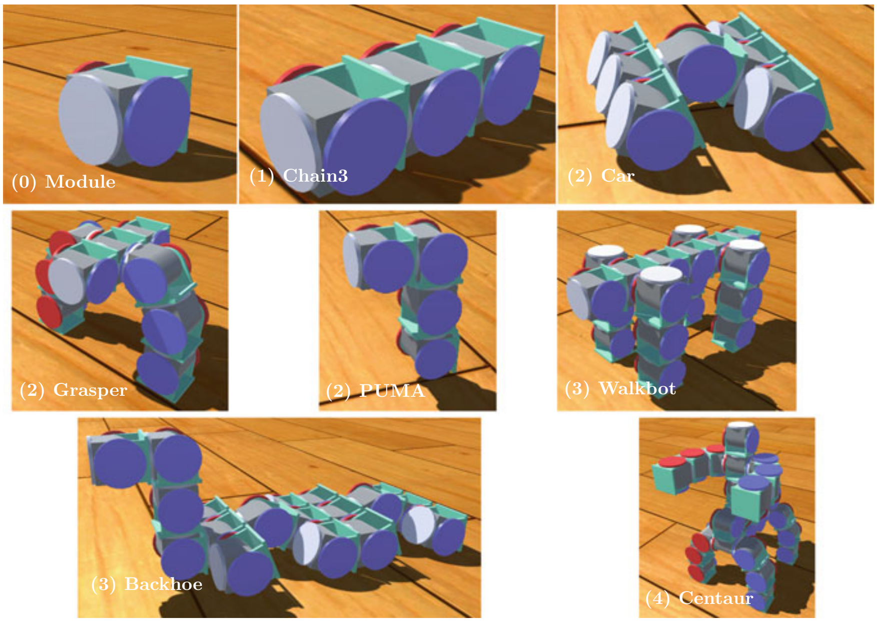


Fig. 7 Library of designs, listed by order. **Order 0:** Module. **Order 1:** Chain3 = 3x Module. **Order 2:** Car = Chain3 + 4x Module, Grasper = 3x Chain3, PUMA = Chain3 + Module. **Order 3:** Walkbot = 2x Grasper, Backhoe = 2x Car + PUMA. **Order 4:** Centaur = Walkbot + Grasper + 2x Module

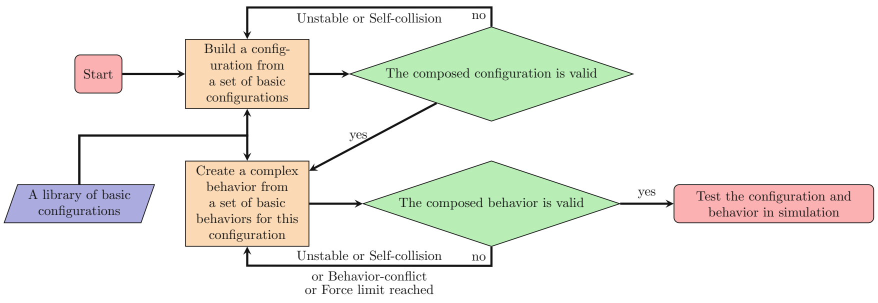

Start
Build a configuration from a set of basic configurations
The composed configuration is valid
Unstable or Self-collision
no
yes
A library of basic configurations
Create a complex behavior from a set of basic behaviors for this configuration
The composed behavior is valid
yes
Test the configuration and behavior in simulation
Unstable or Self-collision
no
or Behavior-conflict
or Force limit reached

Fig. 8 The design flow

Considering the small library in Fig. 7, it is interesting to note that a diverse set of second- and third-order configurations can be constructed from only one zero- and one first-order configuration. Developing metrics to evaluate the quality of such a library is an interesting opportunity for future work.

Figure 8 demonstrates the design flow when a user is designing a configuration and its behaviors. We present the start-to-end user perspective in designing a complicated configuration called Centaur. Consider the order-1 “Chain3” configuration. A second-order “Grasper” configuration, capable of grasping objects, can be formed

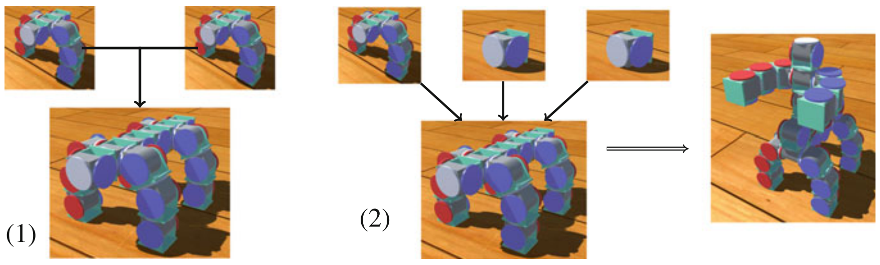


Fig. 9 Building the Centaur. **a** Two Graspers are composed to form a Walkbot. **b** The Walkbot is composed with one more Grasper and two individual modules to form the Centaur

by combining three first-order “Chain3” configurations. Combining two Graspers allows us to form the legs and body of the third-order “Walkbot” configuration, now using the Grasper arms as legs for walking, as demonstrated in Fig. 9a. If we attach another Grasper to the top of the Walkbot (with two additional modules for structural support), we get the fourth-order “Centaur”, a mobile manipulator, as shown in Fig. 9b. Connecting multiple lower-order configurations allows us to quickly develop complex high-order configurations like the Centaur. Given access to a library already containing the Grasper design, for example, creating the Centaur is involves just two composition steps. The user can then immediately compose behaviors already associated with the lower-order configurations (like “Walk” and “Grasp”) to create behaviors for higher-order configurations (like picking up and carrying an object).

7 Conclusions

In this paper, we presented a design framework that facilitates the rapid creation of configurations and behaviors for modular robots. Complex configurations are hierarchically constructed from basic subcomponents. We presented a novel motion description language, which allows existing behaviors to be combined in series and parallel to create more complex behaviors. The framework verifies configurations and behaviors, allowing early detection of design flaws, specifically behavior conflict, self-collision, loss of gravitational stability, and forces exceeding the limits of safety for actuators and connectors. In addition to verification, designs can be evaluated in a physical simulator before testing on hardware.

8 Future

Future work will include expansion of the features of our framework. We hope to expand the capabilities of our motion description language without sacrificing ease-of-use, and add more verification tools to assist in problem identification. Another area of future work lies in developing a standard library of configurations and behaviors for the SMORES robot. We will also investigate metrics to evaluate the quality of such a library. Perhaps most importantly, we will test and evaluate the designs and behaviors with actual hardware modules. Currently, each behavior is associated with exactly one configuration. In many cases, a given behavior could be executed by several different configurations (if it were correctly mapped onto a subset of their modules). In the future, we will apply an embedding detection algorithm (see [12]) to map behaviors into any configuration capable of executing them.

Finally, while our implementation currently supports only SMORES, other modular robots could be easily incorporated. In the future, we plan to incorporate support for the CKbot robot [4] into our software.

References

1. Bezzo, N., et al.: Demo abstract: Roslaba modular programming environment for robotic applications. In: ICCPS (2014)
2. Casal, A.: Reconfiguration planning for modular self-reconfigurable robots. Ph.D. Thesis, Stanford Univ. (2001)
3. Christensen, D., Brandt, D., Stoy, K., Schultz, U.P.: A unified simulator for self-reconfigurable robots. In: IROS (2008)
4. Davey, J., Sastra, J., Piccoli, M., Yim, M.: Modlock: a manual connector for reconfigurable modular robots. In: IROS (2012)
5. Davey, J., et al.: Emulating self-reconfigurable robots: design of the SMORES system. In: IROS (2012)
6. Fukuda, T., Kawauchi, Y.: Cellular robotic system (CEBOT) as one of the realization of self-organizing intelligent universal manipulator. In: ICRA (1990)
7. Hornby, G.S., Lipson, H., Pollack, J.B.: Generative representations for the automated design of modular physical robots. In: ICRA (2003)
8. Huang, J., et al.: ROSRV: Runtime verification for robots. In: Runtime Verification, pp. 247–254. Springer (2014)
9. Kasper Stoy David Brandt, D.J.C.: Self-reconfigurable robots: an introduction. MIT Press, Cambridge (2010)
10. Koenig, N., Howard, A.: Design and use paradigms for gazebo, an open-source multi-robot simulator. In: IROS (2004)
11. Lipson, H., Pollack, J.B.: Towards continuously reconfigurable self-designing robotics. In: ICRA (2000)
12. Mantzouratos*, Y., Tosun*, T., Khanna Sanjeev, Y., Mark: On embeddability of modular robot designs. In: ICRA (2014)
13. Marzinotto, A., Colledanchise, M., Smith, C., Ogren, P.: Towards a unified behavior trees framework for robot control. In: ICRA (2014)
14. Mehta, A., et al.: A design environment for the rapid specification and fabrication of printable robots. In: ISER (2014)
15. Pan, J., Chitta, S., Manocha, D.: FCL: a general purpose library for collision and proximity queries. In: ICRA (2012)
16. Physx. http://www.geforce.com/hardware/technology/physx. Accessed: 2015-04-35
17. Quigley, M., Conley, K., Gerkey, B.P., Faust, J., Foote, T., Leibs, J., Wheeler, R., Ng, A.Y.: ROS: an open-source robot operating system. In: ICRA Workshop on Open Source Software (2009)
18. Salemi, B., Shen, W.M., Will, P.: Hormone-controlled metamorphic robots. In: ICRA (2001)

19. Sklyarov, V., Skliarova, I.: Design and implementation of parallel hierarchical finite state machines. In: Second International Conference on Communications and Electronics, ICCE 2008, pp. 33–38. IEEE (2008)
20. Sproewitz, A., Moeckel, R., Maye, J., Ijspeert, A.J.: Learning to move in modular robots using central pattern generators and online optimization. Int. J. Robot. Res. **27**(3–4), 423–443 (2008)
21. Stoy, K., Shen, W.M., Will, P.M.: Using role-based control to produce locomotion in chain-type self-reconfigurable robots. IEEE/ASME Trans. Mech. **7**, 410 (2002)
22. Unity3d. http://unity3d.com/. Accessed: 2015-04-35
23. Valdes, J., Tarjan, R.E., Lawler, E.L.: The recognition of series parallel digraphs. In: Proceedings of the ACM Symposium on Theory of Computing (1979)
24. Walter, J.E., Tsai, E.M., Amato, N.M.: Choosing good paths for fast distributed reconfiguration of hexagonal metamorphic robots. In: ICRA (2002)
25. Yim, M.: Locomotion with a unit-modular reconfigurable robot. Ph.D. Thesis, Stanford (1994)
26. Yim, M., Duff, D.G., Roufas, K.: Modular reconfigurable robots, an approach to urban search and rescue. In: 1st International Workshop on Human-friendly Welfare Robotic Systems (2000)
27. Yim, M., Duff, D.G., Roufas, K.D.: PolyBot: a modular reconfigurable robot. In: ICRA (2000)
28. Yim, M., et al.: Modular self-reconfigurable robot systems. IEEE Robot. Autom. Mag. **14**, 43 (2007)
29. Yoshida, E., et al.: A self-reconfigurable modular robot: reconfiguration planning and experiments. In: IJRR (2002)
30. Zhang, Y., et al.: Phase automata: a programming model of locomotion gaits for scalable chain-type modular robots. In: IROS (2003)

Automated Fabrication of Foldable Robots Using Thick Materials

Cynthia Sung and Daniela Rus

1 Introduction

Designing complex machines such as robots often requires multiple iterations of design and prototyping. In addition, fabricating a robot using traditional manufacturing techniques involving machining specialized parts and attaching them together can be a long and arduous process, meaning that the design cycle itself can also take a long time.

Recent interest in accelerating fabrication has led to a wide variety of techniques in 3D printing of entire mechanisms [4, 23], molding and casting [12], and laser cutting components for assembly [2, 3], among others. Assembly via folding [9, 14, 18] allows entire robots to be created in several hours. Furthermore, since all fabrication prior to folding is planar, circuitry and actuators can easily be incorporated into the robot body [1, 16].

Using folding for assembly introduces additional constraints on the robot that designers must take into account during the design process. As a result, composition has emerged as a method for designing folded structures. Many foldable modules have been designed [7, 11, 20, 22] that can be combined with each other in order to produce complex mechanisms. For example, work in [22] proposed a continuously foldable cylinder that could be tessellated to produce cellular structures that expand and contract. Work in [7, 20] introduced new joint types that spanned many of the joints used in conventional robots.

In [17], we developed a data-driven system that guides users through the design composition process. Users can choose components from a database of foldable robot parts and combine them with each other to produce custom robot designs. As the

C. Sung (✉) · D. Rus
Massachusetts Institute of Technology, Cambridge, MA 02139, USA
e-mail: crsung@csail.mit.edu

D. Rus
e-mail: rus@csail.mit.edu

A. Bicchi and W. Burgard (eds.), *Robotics Research*, Springer Proceedings in Advanced Robotics 2, DOI 10.1007/978-3-319-51532-8_16

user interacts with the system, the system ensures that the robot remains fabricable (i.e., that the fold pattern is valid and printable). Using examples from the database, it recommends a motion sequence for the articulated joints that allows the robot to move forward, and it suggests changes to dimensions that improve the robot's stability. Once the user is satisfied with the design, the system outputs a 3D mesh that can be printed using a 3D printer and folded into the robot body.

The system in [17] uses a combination of 3D printing and folding to enable rapid fabrication of lightweight robots. However, the resulting designs are limited in size to what can fit in the 3D printer and are intended to have thin walls that do not interfere with the printed fold, so they are limited in the types of tasks that they can accomplish. In this paper, we extend the system to enable fabrication of foldable robots that can be deployed for tasks requiring larger size and higher strength. Since the system ensures the validity of a fold pattern and generates a 3D printable mesh as a separate step, it is not actually specific to 3D printing. The same fold pattern can be implemented using any method that can generate the necessary folds and joints for the pattern and that produces a robot that can maintain its shape.

A larger robot that can carry a larger payload requires a body made of stiffer material. At the same time, we would like the robot to remain lightweight and quickly fabricable. We introduce a new fabrication option that involves laser cutting thick materials to produce larger, more rigid robot bodies than those 3D printed previously (Ref. Fig. 1). Previous work [15, 25] has shown that it is possible to fold a plastic sheet into a rigid body by heating up the material along the fold line. However, these approaches are limited by how quickly the material can be heated and cooled and can only produce objects with one-piece fold patterns. We instead cut rigid faces and layer them on top of a flexible film. The film enables folding similarly to work in [11, 13, 16, 20], and the added rigid faces maintain high stiffness in the robot body. Our techniques are related to previous work in thick origami [21, 24, 26], which also use layers to produce rigidly foldable structures. However, compared to traditional origami fold patterns, which contain many cycles of connected faces that can help the structure maintain its shape, the fold patterns for foldable robots and mechanisms

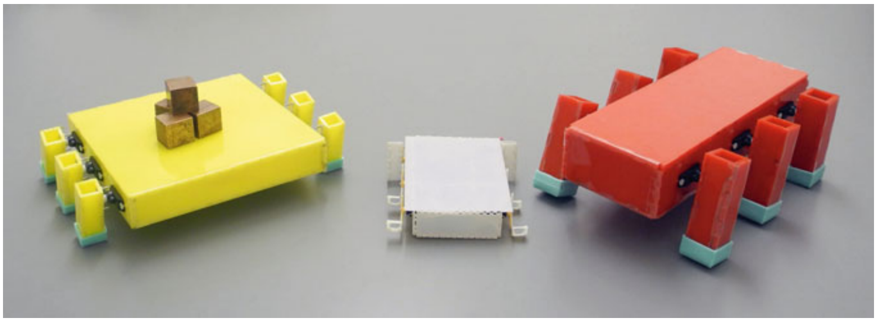


Fig. 1 Hexapods cut from thick material next to 3D printed hexapod (*middle*) fabricated in [17]. The new hexapods are almost twice as large

are often simpler and more treelike. We therefore introduce interlocking teeth along the edges of joined faces to prevent slip.

This paper contributes the following:

- a fabrication process for foldable robots that involves cutting and layering a rigid, thick material for stiffness and flexible film to allow folding,
- an algorithm for automatically generating cut patterns for the layered structure given the fold pattern for a robot design, and
- experimental verification for four rigid shapes and two hexapods of different sizes that were fabricated using the proposed process.

This paper is structured as follows. Section 2 summarizes the design process and the information associated with a folded robot design. Section 3 outlines our fabrication and assembly process, and Sect. 4 describes how the fabrication plans can be automatically generated from the design information in Sect. 2. In Sect. 5, we demonstrate the structures that were created using the proposed method, and we conclude in Sect. 6.

2 Parametric Cut-and-Fold Design

Our folded designs are composed as described in [17]. The system allows users to compose and customize new ground robots while ensuring fabricability and stable forward locomotion. Since we fabricate robots using a cut-and-fold process, the 3D geometry of the robot is designed concurrently with the 2D fold pattern.

The 3D geometry and the fold pattern each consist of polygonal faces that are joined along edges. There is a one-to-one correspondence between the faces of the 3D geometry and the faces of the fold pattern. We say that edges joining faces that are adjacent in the fold pattern are themselves adjacent. Nonadjacent joining edges are edges that join two faces in the 3D geometry but are not adjacent in the fold pattern. Both types of joining edges are annotated with the dihedral angle between the two faces in the 3D geometry.

The designs are parametrized so that users can first conceive the general shape of the robot and subsequently optimize over the size and shape to achieve stable forward locomotion. For example, the hexapod pictured in Fig. 2 has parameters of body length, body width, body height, leg length, and leg width. Updates to the parameters of the robot simultaneously change the 3D geometry and the fold pattern. This approach introduces some constraints on the parameters, such as not allowing self-intersection between faces of the fold pattern, that are checked automatically by the system.

Finally, the system suggests a motion sequence for each articulated joint in the design that will allow the robot to locomote forward, and it simulates the motion of the robot to ensure that the robot maintains static stability throughout the motion sequence. The system provides suggestions for parameter changes if the designed

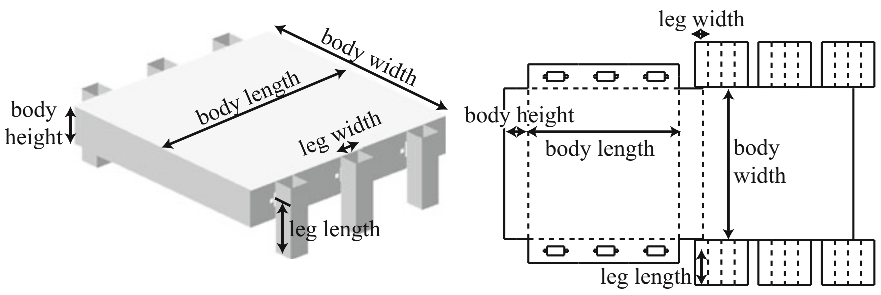


Fig. 2 Hexapod geometry and corresponding fold pattern with parameters labeled

robot is not stable. The user of the system can continue to modify the design by changing parameters or adding and removing parts until satisfied.

3 Fabrication and Assembly

Once the design is finalized, the system automatically converts the fold pattern into something that can be fabricated. We laser cut rigid materials and layer them with a thin film so that they can be folded into the robot body. Electronics and software are also generated for the motion sequences of the articulated joints.

3.1 Robot Body

Since robots folded from flexible materials are unable to carry substantial load [19], we fabricate rigid robots by layering a thick, rigid material and a flexible film in the shape of the fold pattern and then folding the structure into its 3D form. The process is illustrated in Fig. 3. All steps except the final assembly step use planar fabrication.

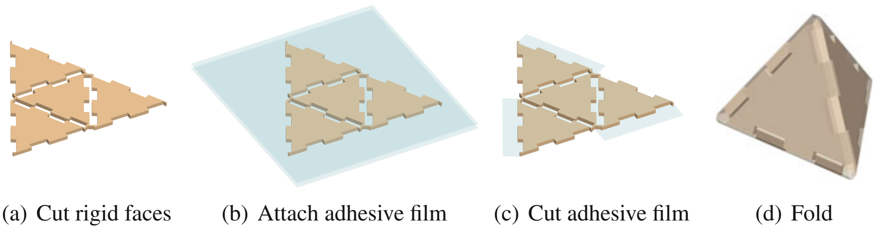


Fig. 3 Fabrication process for the robot body. **a** The fold pattern is first cut from a rigid material. **b** Adhesive film is attached to the surface of the rigid material and **c** then cut into the correct shape. **d** Finally, the layered structure is folded into its final 3D form. Tabs cut from the adhesive film are folded to secure the shape

The faces of the fold pattern are first cut out of a rigid material using a laser cutter. We chose to use acrylic sheet for its high surface energy and for ease of laser cutting. To account for material thickness, the faces are shrunk and separated by a gap, similarly to [21]. While cutting individual faces, thin strips that bridge the gaps are kept so that the structure remains a single piece and the correct gap width is preserved. These strips are later cut during folding. Interlocking teeth along each pair of joining edges add structural integrity and keep faces from sliding along the fold line.

Once the rigid material is cut, a thin, adhesive-backed polyester film is layered on the top and bottom. The film maintains the geometry of the rigid material even once the thin strips between faces have been cut. The film is then cut using a laser cutter. For each fold, either the top or bottom film is cut along the toothed edges and removed, depending on the sign of the fold angle, to allow folding. Since the two films are separated by the thickness of the rigid material, they can be cut individually without affecting the integrity of the other side. For nonadjacent joining edges, we add a tab to one of the pair of edges so that the two can be secured together during folding. Finally, the thin strips connecting the rigid faces are also cut and removed.

During assembly, the cutout is folded into its 3D form. Tabs cut from the adhesive film are wrapped around the corresponding folds to secure the two fold edges together.

3.2 Electronics and Software

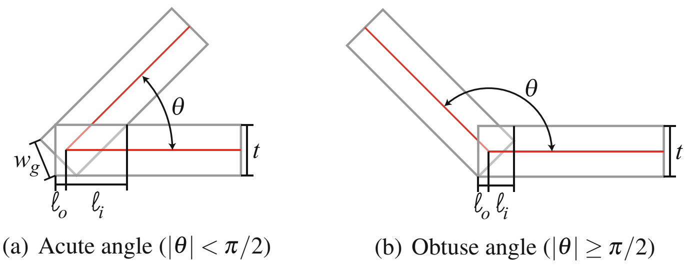
The motion sequence associated with the robot design is translated into actuators and a software controller. This step is currently performed manually since additional information about the voltage and current requirements of circuit components such as the microcontroller and sensors, as well as the control input to the actuators, are not taken in account by the design system. We actuate our robots using servomotors. The times and waypoints of the motion sequence are converted into a PWM (pulse width modulation) signal that is sent to the servomotors using an Arduino Uno.

4 Cut Pattern Generation

The system automatically generates the cut patterns required for fabricating the robot body. The cut pattern consists of two parts: (1) cuts for the thick rigid layer, and (2) cuts for the thin adhesive layers.

4.1 Rigid Material

Three-dimensional bodies with thick walls have faces of different sizes when viewed from one side compared to the other. This creates a challenge for laser cutting the



θ
t
w_g
ℓ_o
ℓ_i
(a) Acute angle ($|\theta| < \pi/2$)
θ
t
ℓ_o
ℓ_i
(b) Obtuse angle ($|\theta| \geq \pi/2$)

Fig. 4 Side view of two faces of thickness t that come together at a fold with angle θ between them. The *midline* of the material, indicated in *red*, shows the original length of the faces. The length ℓ_o is the greatest amount that a face can be lengthened without breaking through the surface of the other face. The length ℓ_i is the length that the face must be shortened to avoid intersection with the other face. The gap width w_g is the amount of space that must be placed between two faces joined at an acute angle once they have been lengthened by ℓ_o

faces of the body since cuts are perpendicular to the faces being cut. As a result, faces cannot always be cut to the same size as they were in the original fold pattern.

Figure 4 shows the resulting geometry for two faces of thickness t that are joined with an angle $\theta \in (-\pi, \pi]$ between them. Because of the thickness of the material, when the two faces are joined at an angle, there exists either some overlap or a gap between the faces. The red line indicates the slice of the material that has the same dimensions as the original fold pattern. This is equivalent to saying that the material was folded at that slice. The location of the slice can be designed to occur at any position within the thickness of the material, or even outside of the material. We choose for it to be at the midline of the material for symmetry.

Faces joined at an edge must be trimmed so that no intersection occurs during folding. We also add teeth along a joining edge so that the joined faces interlock for greater rigidity. We draw rectangular teeth relative to the original location of the edge of the face and refer to the distances of the outermost and innermost points from the edge location as ℓ_o and ℓ_i respectively. The value of ℓ_o can be calculated as the greatest amount that one of the faces can be lengthened without breaking through the surface of the other face. The value of ℓ_i is the amount that the face must be shrunk in order to avoid intersection with the other face, once lengthened. These two values vary depending on the angle between the two joined faces as

$$\ell_i = \begin{cases} \frac{t}{2\tan\frac{\theta}{2}} & |\theta| < \pi/2 \\ t\sin(\theta) - \frac{t}{2\tan\frac{\theta}{2}} & |\theta| \geq \pi/2 \end{cases} \tag{1}$$

$$\ell_o = \begin{cases} \frac{t}{\sin\theta} - \frac{t}{2\tan\frac{\theta}{2}} & |\theta| < \pi/2 \\ \frac{t}{2\tan\frac{\theta}{2}} & |\theta| \geq \pi/2 \end{cases} \tag{2}$$

The cut pattern for the rigid material is drawn by shrinking and shifting the faces and adding teeth to the fold edges. The procedure is as follows:

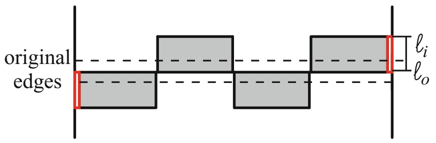
1. **Shrink Faces**. For each pair of joined faces, shrink the face by ℓ_i in the direction perpendicular to the joining edge. This results in a gap of width $2\ell_i$ between the two faces.
2. **Shift Faces**. Shift the faces by an amount depending on the whether the angle between the faces is acute or obtuse. For obtuse angles (Fig. 4b), the faces must be further separated by a distance of $2\ell_o$. For acute angles (Fig. 4a), the faces must be separated by a distance of $2\ell_o + w_g$, where

$$w_g = 2\ \sin\frac{\theta}{2}\left(\frac{t}{\tan\frac{\theta}{2}} - \frac{t}{\sin\theta}\right) \tag{3}$$

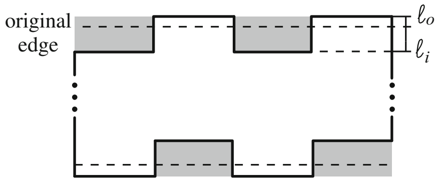
   is the distance across the shortened corner caused by cutting the faces so that they do not extend past the angle formed by the outer surfaces.
3. **Add Teeth**. For each pair of joining edges, add alternating rectangular teeth to the two edges. The teeth extend a width $\ell_i + \ell_o$ past the edges of the shrunken faces. The number of teeth was chosen such that each tooth was at most 10 mm long and there were at least three teeth. In addition, teeth were cut 0.3 mm wider than the gap on the opposite edge to account for the kerf of the laser cutter.
   In order for the material to remain one piece until assembly, thin strips of material that bridge the gaps between adjacent joining edges are kept. This is done by drawing the teeth for a length 2 mm shorter than the actual edge length. Figure 5 shows the resulting teeth structure for both adjacent and nonadjacent joining edges. Red lines on the teeth for adjacent edges show the pieces that are cut off prior to assembly.

4.2 Adhesive Layers

Adhesive layers are placed on both the top and the bottom of the rigid layer and are cut away to allow folding. We use the convention that when the angle θ between two



(a) Adjacent joining edges



(b) Nonadjacent joining edges

Fig. 5 Teeth structure for a pair of edges that form a fold. **a** *Red lines* indicate the pieces that keep the structure connected but are cut prior to assembly

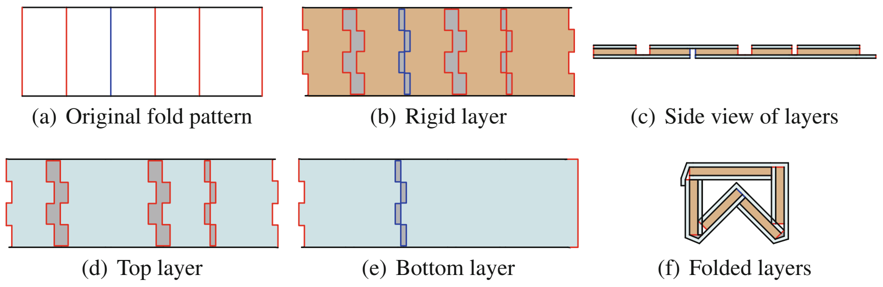


Fig. 6 Adhesive layers for an example fold pattern. **a** Original fold pattern. *Red lines* correspond to joining edges with positive angle, and *blue lines* correspond to negative angle. **b** Cut pattern for rigid layer. **d, e** Resulting cut patterns for *top* and *bottom* adhesive-backed layers. **c, f** Side view of the resulting layered structure

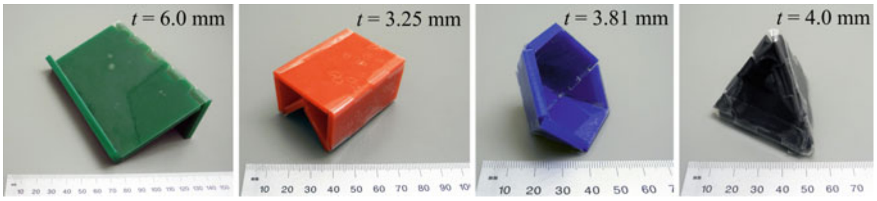
joined faces is positive, then the top layer is cut away (Ref. Fig. 6c); when the angle is negative, then the bottom layer is cut.

To draw the cut pattern of the top layer, we begin with the cut pattern of the rigid material. Then, for each adjacent joining edge, the cut lines corresponding to the teeth are removed if θ is negative and remain unmodified otherwise. For each nonadjacent edge, the cut lines corresponding to the teeth are similarly modified only if θ is negative. In that case, the cut lines corresponding to the teeth are kept unmodified for one edge, and the other edge is replaced by a rectangular tab. The process is repeated in the same fashion for the bottom layer, except that cut lines corresponding to joining edges with negative angle θ are left unmodified instead. Fig. 6 shows the resulting top and bottom layers for an example fold pattern.

5 Results

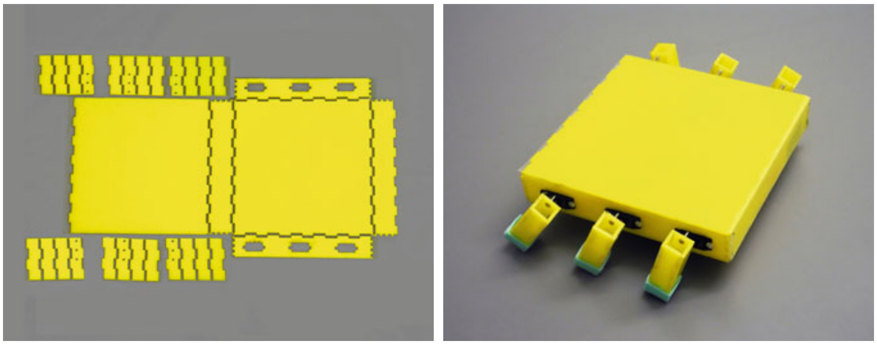
We have used the fabrication process described in Sect. 3 to create various folded structures. Figure 7 shows four of the shapes we made. In order to test the system's ability to generate cut patterns for different thicknesses of material, we used a different thickness of acrylic for each of the shapes. The thicknesses are indicated in the figure. We layered all the structures with a 0.05 mm thick polyester film backed with acrylic adhesive.

The green stand (Fig. 7a) is the simplest structure with both positive and negative degree folds. Because of the adhesive on the polyester film and the fit of the teeth, the angles of the structure remain fixed after folding. The red table structure (Fig. 7b) is the folded strip shown in Fig. 6. It also contains both positive and negative fold angles, as well as a nonadjacent joining edge. The blue half cuboctahedron (Fig. 7c) and black pyramid (Fig. 7d) each have more than two faces joined at a vertex. Because of the teeth design, both of these structures were able to fold without any interference between any of the faces.



(a) Stand (b) Table (c) Half cuboctahedron (d) Tetrahedron

Fig. 7 3D structures cut and folded from acrylic sheets of varying thicknesses (thickness indicated on the image). The structures in **b** and **d** are physical prototypes of the patterns shown in Figs. 6 and 3 respectively



(a) Fold pattern (b) Assembled robot

Fig. 8 Wide hexapod cut out of 3.18 mm thick acrylic sheet using our fabrication process

We also created two hexapods of different dimensions. The pattern for the hexapod consists of a single rectangular body and six rectangular beam legs, shown in Fig. 2. Although the legs are drawn adjacent to the body, they are not connected via any joining edges. The pattern was originally designed for 3D printing [17], but the same fold pattern can be used to create a larger, stronger robot using rigid materials. Again, to test the system's ability to generate cut patterns for different thicknesses of material, we cut a short and wide hexapod (Fig. 8) out of 3.18 mm thick acrylic sheet and a tall and long hexapod out of 4.50 mm thick acrylic sheet (Fig. 9). Both robots were layered with a 0.05 mm thick polyester film backed with acrylic adhesive. The cut patterns for the hexapods both folded into the correct shapes.

The robots were each actuated using six servomotors with stall torques of 2.7 kg-cm and controlled using an Arduino Uno. They were powered by two 3.7 V, 2600 mAH lithium ion batteries regulated to 6 V to meet the power requirements of the servomotors. The servomotors were modified to allow continuous rotation while still providing position feedback from the potentiometer connected to the output shaft. During assembly, the servomotors were screwed into mounting holes designed into the robot body. Similarly, servo horns were screwed to each of the six legs and then snapped onto the shafts of the servomotors.

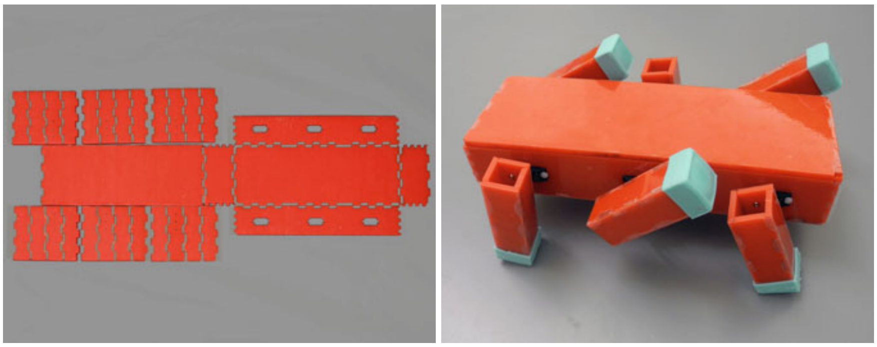

(a) Fold pattern (b) Assembled robot

Fig. 9 Long hexapod cut out of 4.50 mm thick acrylic sheet using our fabrication process

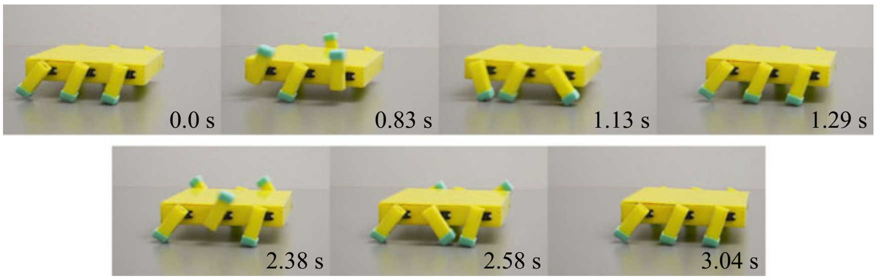


Fig. 10 One cycle in the walking gait of the wide hexapod. Three legs make one full rotation to shift the robot forward while the other three keep the robot stable. Next the other three legs rotate

Both hexapods were programmed to follow the same tripod gait as suggested by the design system. For each of the servomotors, the Arduino Uno received analog input from the potentiometer and outputted a PWM control. The resulting walking gait is shown in Fig. 10. All the legs were initialized to an angle of 20° (0.35 rad) offset from vertical. The legs were then split into two sets of three, with each set containing the front and back legs of one side of the robot and the middle leg of the other. One set of legs rotated one complete revolution in 1.5 s while the other set remained static to maintain the stability of the robot. The sets of legs alternated between rotating and remaining static.

Table 1 shows the amount of time required to fully fabricate and assemble each robot. Fabrication of the layered structure took approximately 1 h for each robot, and full assembly, including attaching electronics, took an additional 1.5 h. The long hexapod, which is the larger and heavier of the two, took more time to fabricate and assemble than the wide hexapod. Since it is longer, the entire fold pattern did not fit on one sheet of acrylic, so the robot was fabricated in two parts that were attached together using the adhesive-backed film. This process doubled the amount of time

Table 1 Timing per step of fabrication

Timing	Wide hexapod	Long hexapod
Cut rigid layer	10 min	20 min
Attach adhesive layer	25 min	50 min
Cut adhesive layer	5 min	5 min
Assemble	45 min	60 min
Attach electronics	25 min	25 min
Total	1 h 50 min	2 h 40 min
3D print body	10 h 56 min	14 h 26 min
3D print faces	9 h 28 min	12 h 25 min

required to attach the adhesive layer to the rigid layer for the long hexapod compared to the wide hexapod.

Rubber feet were placed on each of the legs to prevent slip during the walking gait. The feet were fabricated by 3D printing a mold, pouring A15 durometer silicone rubber into the mold, and then allowing the rubber to cure. The entire process took about 5 h.

Compared to 3D printing the hexapods, laser cutting and folding the robots resulted in substantial time and materials savings. In particular, 3D printing the long hexapod takes almost 14.5 h on a Fortus 400mc printer (Table 1), and 1/4 of the printed material is support for the hollow body. Even separating the body into individual faces to minimize support material still requires 12.5 h of printing, not to mention the additional time for assembly. Similarly, printing the wide hexapod takes 11 h, with 1/3 of the material being support material, or 9.5 h for printing just the faces.

Both robots were able to walk forward stably, although lack of synchrony between the servomotors occasionally caused the robots to shuffle and turn during steps. During experiments, the robots turned at most 10° (0.17 rad) per step in the direction of the side whose middle leg was moving. Table 2 describes the size and performance

Table 2 Robot specifications

	Wide hexapod	Long hexapod
Thickness	3.18 mm	4.50 mm
Length	192 mm	288 mm
Width	260 mm	190 mm
Height	60 mm	95 mm
Weight	0.672 kg	1.058 kg
Speed	27.7 mm/s	35.9 mm/s
Payload capacity	2.50 kg	0.76 kg

of each robot. Speed was computed by measuring the amount of time required for the robot to walk forward 3 m and averaging the result over 3 trials. Payload capacity was measured by incrementally adding weights on top of the robot until its leg sets were no longer able to complete a full rotation (i.e., the robot could no longer walk forward).

As might be expected, the long hexapod could move faster than the wide hexapod but was able to carry less payload. While the wide hexapod was able to carry more than 3.5 times its own body weight, the long hexapod, which uses the same servomotors but is taller and heavier, was able to carry only about 0.72 times its own weight. In terms of speed, both hexapods used the same motion sequence, but the long hexapod has longer legs and so moved about 1.30 times as fast as the wide hexapod.

6 Discussion

In this paper, we describe and demonstrate a method of rapid fabrication that can be used to create foldable robots. Previous work in designing folded robots often assumes that the material is infinitely thin, resulting in physical implementations that can not sustain much load [19]. We use a combination of a thick, rigid material and an adhesive-backed film to create a layered structure that enables folding while maintaining rigidity in the faces of the robot body. We incorporate thickness considerations into the design process, and we show how to automatically generate the cut lines for the layers forming the foldable robot. We have verified our process by fabricating multiple structures with different dimensions and using different thicknesses of material. Our results indicate that folding may be a viable method for quickly prototyping robots that must also complete everyday tasks.

One limitation of our approach is that the folding occurs off axis, meaning that there is a minimum angle between faces that can be achieved. In particular, angle values close to zero will result in a large amount of shrinking for the corresponding faces and may be infeasible using the proposed approach. However, several instances of previous work [5, 10] have considered thick materials in the context of flat folding (i.e., angle of 0 between faces). We hypothesize that many of these methods could be combined with our teeth structure to allow sharp angles in a folded structure. We also plan to perform more experiments to assess the structural integrity of the folded robot structures using our fabrication method as compared to previous thick folding work and 3D printing.

Secondly, as mentioned in [21], shifting faces to make room for joining edges could cause self-intersection in the resulting cut pattern or yield inconsistencies if the fold pattern contains cycles. Since most of our robot designs are simple trees, we have not yet encountered these kinds of issues. However, we foresee having to solve this problem as the robots become more complex.

One of the advantages of 3D printing for smaller robots [17] was that complex 3D joints could be incorporated into the design without adding additional complexity to the fabrication process. There exists a large collection of patterns that can be used to create joints from flat and rigid sheets [6, 8]. In the future, we would like to investigate how to incorporate such joints into our fabrication process.

Finally, all of the electronics and software for the robots were created manually. In particular, stronger servomotors were used compared to the smaller robots produced for [17]. This resulted in higher voltage and current requirements, which necessitated more complex circuitry. Work such as that in [13] indicates that the electronics, software, and mechanical body of a robot can be simultaneously designed automatically. Future work includes incorporating information about electronics and software into our design system and fabrication process to enable simple design, customization, and fabrication of a robot in its entirety.

Acknowledgements Support for this project has been provided in part by NSF Grant Nos. 1240383 and 1138967, and by the DoD through the NDSEG Fellowship Program. We are grateful. We would also like to thank John Romanishin and Joseph DelPreto for helpful discussions, and Bianca Homberg for assistance in fabricating the silicone feet.

References

1. An, B., Rus, D.: Designing and programming self-folding sheets. Robot. Auton. Syst. **62**(7), 976–1001 (2014)
2. Beyer, D., Gurevich, S., Mueller, S., Chen, H.T., Baudisch, P.: Platener: low-fidelity fabrication of 3D objects by substituting 3D print with laser-cut plates. In: Proceedings of ACM Conference on Human Factors in Computing Systems (2015)
3. Chen, D., Sitthi-amorn, P., Lan, J.T., Matusik, W.: Computing and fabricating multiplanar models. Comput. Gr. Forum **32**, 305–315 (2013)
4. Coros, S., Thomaszewski, B., Noris, G., Sueda, S., Forberg, M., Sumner, R.W., Matusik, W., Bickel, B.: Computational design of mechanical characters. ACM Trans. Gr. **32**(4), 83 (2013)
5. Edmondson, B.J., Lang, R.J., Magleby, S.P., Howell, L.L.: An offset panel technique for thick rigidily foldable origami. In: Proceedings of ASME International Design Engineering Technical Conferences and Computers and Information in Engineering Conference (2014)
6. Fenner, P.: Laser-cut lattice living hinges. http://www.deferredprocrastination.co.uk/blog/2011/laser-cut-lattice-living-hinges/ (2011)
7. Gao, W., Ramani, K., Cipra, R.J., Siegmund, T.: Kinetogami: a reconfigurable, combinatorial, and printable sheet folding. J. Mech. Design **135**(11), 111009 (2013)
8. Goldberg, S.A.: Designing continuous complex curved structures to be fabricated from standard flat sheets. In: Vision and Visualization: Proceedings of the 9th Iberoamerican Congress of Digital Graphics, pp. 114–119 (2005)
9. Hoover, A.M., Steltz, E., Fearing, R.S.: RoACH: An autonomous 2.4g crawling hexapod robot. In: Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems, pp. 26–33 (2008)
10. Ku, J., Demaine, E.: Folding flat crease patterns with thick materials. In: Proceedings of ASME International Design Engineering Technical Conferences and Computers and Information in Engineering Conference (2015)
11. Lee, D., Kim, J., Kim, S., Koh, J., Cho, K.: The deformable wheel robot using magic-ball origami structure. In: Proceedings of ASME International Design Engineering Technical Conferences and Computers and Information in Engineering Conference, pp. DETC2013–13016 (2013)

12. Ma, R.R., Belter, J.T., Dollar, A.M.: Hybrid deposition manufacturing: design strategies for multi-material mechanisms via three-dimensional printing and material deposition. J. Mech. Robot. **7**, 021002 (2015)
13. Mehta, A., DelPreto, J., Rus, D.: Integrated codesign of printable robots. J. Mech. Robot. **7**, 021015 (2015)
14. Mehta, A., Rus, D.: An end-to-end system for designing mechanical structures for print-and-fold robots. In: Proceedings of IEEE International Conference on Robotics and Automation (2014)
15. Mueller, S., Kruck, B., Baudisch, P.: LaserOrigami: laser-cutting 3D objects. In: Proceedings of ACM Conference on Human Factors in Computing Systems, pp. 2585–2592 (2013)
16. Niiyama, R., Rus, D., Kim, S.: Pouch motors: printable/inflatable soft actuators for robotics. In: Proceedings of IEEE International Conference on Robotics and Automation (2014)
17. Schulz, A., Sung, C., Spielberg, A., Zhao, W., Cheng, Y., Mehta, A., Grinspun, E., Rus, D., Matusik, W.: Interactive robogami: data-driven design for 3D print-and-fold robots with ground locomotion. In: ACM SIGGRAPH Talks (2015)
18. Soltero, D.E., Julian, B.J., Onal, C.D., Rus, D.: A lightweight modular 12-DOF print-and-fold hexapod. In: Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems, pp. 1465–1471 (2013)
19. Sung, C., Rus, D.: Foldable joints for foldable robots. In: International Symposium on Experimental Robotics (2014)
20. Sung, C., Rus, D.: Foldable joints for foldable robots. J. Mech. Robot. **7**(2), 021012 (2015)
21. Tachi, T.: Rigid-foldable thick origami. Origami **5**, 253–264 (2011)
22. Tachi, T., Miura, K.: Rigid-foldable cylinders and cells. J. Int. Assoc. Shell Spat. Struct. (IASS) **53**(4), 217–226 (2012)
23. Thomaszewski, B., Coros, S., Gauge, D., Megaro, V., Grinspun, E., Gross, M.: Computational design of linkage-based characters. ACM Trans. Gr. **33**(4), 64 (2014)
24. Tolley, M.T., Felton, S.M., Miyashita, S., Xu, L., Shin, B., Zhou, M., Rus, D., Wood, R.J.: Self-folding shape memory laminates for automated fabrication. In: Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems, pp. 4931–4936 (2013)
25. Yasu, K.: MOR4R: Microwave oven recipes for resins. In: ACM SIGGRAPH Talks (2015)
26. Zirbel, S.A., Lang, R.J., Thomson, M.W., Sigel, D.A., Walkemeyer, P.E., Trease, B.P., Magleby, S.P., Howell, L.L.: Accommodating thickness in origami-based deployable arrays. J. Mech. Design **135**(11), 111005 (2013)

Design, Sensing, and Planning: Fundamentally Coupled Problems for Continuum Robots

Arthur W. Mahoney, Trevor L. Bruns, Ron Alterovitz and Robert J. Webster III

1 Introduction

Continuum robot systems have been created for use in a large variety of practical applications, including underwater manipulation, nuclear reactor repair, sanding, spray painting, and medical applications [9, 47]. A variety of continuum-robot devices exist, include multi-backbone devices [48], tendons routed around a backbone [30], concentrically-nested elastic tubes [12, 31], pneumatic actuators [18], shape-memory-based designs [1, 33], and combinations thereof. Some of these are illustrated in Fig. 1a–d. Each type of continuum robot presents a variety of physical design parameters, including diameter, segment number and length, material properties, and actuation systems, among others [47].

The typical process for designing and using a continuum robot in a new application typically involves three discrete steps that are addressed serially. First, one designs the physical robot to be suitable for the task, i.e. able to reach the desired workspace, carry the desired payload, etc. Attention is then turned toward sensing. Sensors are

This material is based upon work supported by the National Institutes of Health under awards R21 EB017952 and R01 EB017467, and the National Science Foundation under award IIS-1054331.

A.W. Mahoney (✉) · T.L. Bruns · R.J. Webster III
Department of Mechanical Engineering, Nashville, TN 37235, USA
e-mail: art.mahoney@vanderbilt.edu

T.L. Bruns
e-mail: trevor.l.bruns@vanderbilt.edu

R.J. Webster III
e-mail: robert.webster@vanderbilt.edu

R. Alterovitz
Department of Computer Science, University of North Carolina, Chapel Hill, NC 27599, USA
e-mail: ron@cs.unc.edu

selected, which may be integrated into the robot, or located off-board. Lastly, given the physical robot, the motion of the robot is planned to accomplish the desired task. In what follows, we briefly discuss the state-of-the-art in these areas and describe the coupling between design, sensing, and planning.

Research on physical robot design has focused largely on developing general purpose manipulators of various types [16, 47]. Intense design effort has been focused on a wide variety of medical applications, which has resulted in many new physical continuum robot structures [9], including concentric precurved tubes [13], which will be used as an example later in this paper. Computational design algorithms are also increasingly being used in the design of concentric-tube robots [4, 8, 40], and similar techniques could be applicable to task-specific design of other types of continuum robot.

Selecting and integrating sensors in continuum robots can be challenging. Often, most of the continuum robot's volume is reserved for actuation, and little room is left for sensing – particularly in the small diameter designs used in medical applications. Many sensing methods have been developed that measure the pose at a discrete points on the continuum robot's backbone such as electromagnetic trackers [17], mechanical strain of the backbone such as the fiber Bragg gratings [29, 32], and the internal moments of a continuum robot's backbone [49]. To keep the continuum-robot diameter small, these sensors are integrated into the robot's mechanical structure when possible. Vision-based methods are also possible and X-ray/CT-imaging [21], ultrasound approaches [19, 23], and optical cameras [5, 15] have been applied. Some of the above sensing systems, which can be used to detect the shape/states of a continuum robot, are illustrated in Fig. 1f–i.

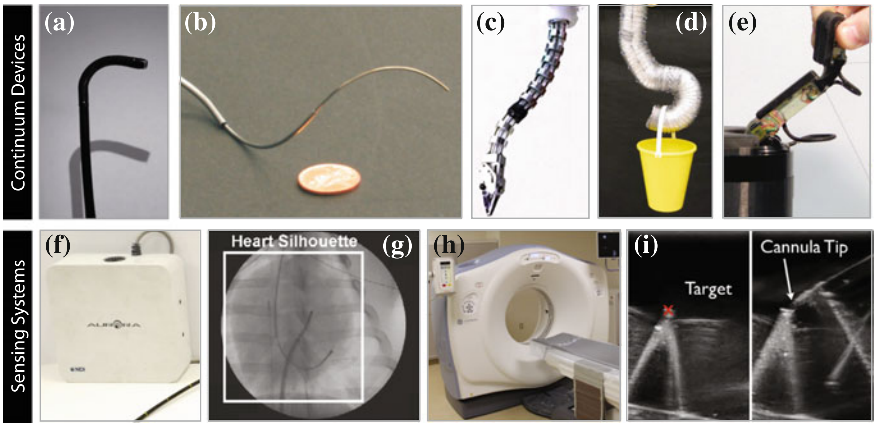


Fig. 1 Continuum robots include devices such as **a** tendon-actuated endoscopes, **b** concentric-tube robots, **c** pushrod-actuated devices, e.g., [48], **d** pneumatic robots, e.g., [18], **e** and some compliant graspers may also fall under this classification, such as that of [24]. Some approaches to sensing the state of continuum robots include **a** magnetic tracking such as the NDI Aurora system, **g–h** X-ray or CT imaging such as [19], and **i** three-dimensional (3D) ultrasound, e.g., [37]

Planning the motion of a continuum robot can enable the robot to automatically reach a specified target while avoiding obstacles in the environment. Motion planning for continuum robots such as concentric-tube robots is challenging because, compared to traditional multi-link manipulators, their kinematics are typically more expensive to evaluate and their motion is subject to substantial uncertainty. Motion planning algorithms for concentric-tube robots include fast motion planning by assuming simplified kinematics [22, 43], noninteractive motion planning using accurate kinematic models [39], interactive-rate motion planning using accurate kinematic modeling via precomputation [41], and motion planners that explicitly consider uncertainty in motion and sensing [36]. Motion planners can also assist in the context of teleoperation, where a fast, real-time motion planner can automatically move the robot's end-effector in response to user commands in a manner that ensures the entire curvilinear shaft avoids obstacles [42].

Recent research in continuum robotics is showing that there are advantages to solving sensing-and-planning problems and design-and-sensing problems simultaneously. In the case of motion planning for continuum robots, for example, effective obstacle avoidance requires accurate estimation of the shape of the robot, and simultaneously planning the motion of the robot and the placement of moveable sensors has the potential to improve task success rates [44]. Motion planning can also help inform the design of concentric-tube robots, e.g., by identifying stable configurations [3] and in optimizing the design of concentric tubes [2, 40].

In this paper argue that design, sensing, and planning are fundamentally coupled problems for continuum robots, and that the interaction between these three problems can be understood through the application of statistical state estimation.

2 Kinematics of a Continuum Robot

We use a notation where scalars are denoted by lower-case standard font (e.g., s), vectors are denoted by bold, lower-case fonts (e.g., $\boldsymbol{x}$), matrices are upper-case standard-font (e.g., M). For compactness, we use subscript notation to denote function arguments. For example, if a vector is a function of s, then it is written as $\boldsymbol{x}_s$.

The kinematics of a continuum robot describe both the continuous spatial transition of the states of the robot's backbone, parameterized by the distance s along the robot's backbone, as well as the continuous temporal transition of the states as an input varies in time. For this present work, we only consider the spatial kinematics that govern how the states $\boldsymbol{x}_s$, parameterized by arc-length distance s, vary along the robot's backbone. We assume that the arc-length distance s falls in the range $[0, \ell]$, where $s = 0$ denotes the proximal end of the continuum robot (i.e., the base) and $s = \ell$ denotes the distal end (i.e., the tip). The state of a continuum robot can include, for example, internal torsional moments and pose of the backbone in $SE(3)$, all of which vary continuously along the length of the backbone with the parameter s. The backbone may be a physical structure as is the case for concentric-tube robots, and some tendon actuated continuum robots, or the backbone could be a theoretical curve used as a reference location with respect to the robot's body.

If the continuum robot is in equilibrium (i.e., quasi static), then the spatial kinematics of a continuum robot can be described by the two-point boundary-value differential equation in arc-length s along the the robot's body:

$$\boldsymbol{x}_s' = \boldsymbol{f}\left(\boldsymbol{x}_s, s\right), \tag{1}$$

where $'$ is an arc-length differentive, and with constraints and inputs of the form

$$\boldsymbol{b}_0(\boldsymbol{x}_0) = \boldsymbol{0} \qquad \boldsymbol{b}_\ell(\boldsymbol{x}_\ell) = \boldsymbol{0} \qquad \boldsymbol{u}(\boldsymbol{x}_0) = \boldsymbol{0} \tag{2}$$

where $\boldsymbol{b}_0$ and $\boldsymbol{b}_\ell$ define the stationary proximal and distal boundary conditions, respectively, and $\boldsymbol{u}$ defines the system input. The proximal boundary condition can constrain properties such as the proximal pose in $SE(3)$ of a continuum robot's body. The distal boundary conditions can constrain properties such as the torsional moment of a concentric-tube robot's tubes, and the net force and moment applied to the end-effector platform of a parallel continuum manipulator by its rod-actuators [7]. The input function $\boldsymbol{u}(\boldsymbol{x}_0)$ operates on the proximal state and can define the proximal tendon displacement of a tendon-actuated device or the proximal twist angles of the tubes comprising a concentric-tube robot.

In this paper, we use concentric-tube robots as examples. Concentric-tube robots, shown in Fig. 1b, consist of precurved, elastic tubes that move in a tentacle-like fashion when the tubes are rotated and translated relative to each other. The states of an unloaded concentric-tube robot are the tubes' twist angles $\boldsymbol{\psi}_s$, their arc-length rates-of-change $\boldsymbol{\psi}_s'$, the arc-length position of each tube $\boldsymbol{\sigma}_s$ relative to the tubes' proximal ends, and backbone position $\boldsymbol{p}_s$ and orientation R_s.

The kinematics of an n-tube concentric-tube robot is governed by

$$\begin{aligned} {}^i\psi_s'' &= -\boldsymbol{u}_s^{\mathsf{T}}\,{}^iK_s\left(\partial R({}^i\psi_s)\right){}^i\boldsymbol{u}_s^* && \text{(3a)}\\ {}^i\sigma_s' &= 1 && \text{(3b)}\\ \boldsymbol{p}_s' &= R_s\boldsymbol{z} && \text{(3c)}\\ R_s' &= R_s S(\boldsymbol{u}_s) && \text{(3d)} \end{aligned}$$

where $\boldsymbol{u}_s$ is the frame curvature, the bending and torsional stiffnesses of tube i are packed in the diagonal matrix ${}^iK_s$, the matrix $\partial R({}^i\psi_s) = \partial R({}^i\psi_s)/\partial{}^i\psi_s$ where $R({}^i\psi_s)$ is the standard z-axis rotation matrix by the angle ${}^i\psi_s$, ${}^i\boldsymbol{u}_s^*$ is the tube precurvature, and $S(\boldsymbol{u}_s)$ is the skew-symmetric matrix representing the cross-product operation. Further details can be found in [12, 31].

The proximal boundary conditions are

$$\boldsymbol{p}_0 = \boldsymbol{0}, \quad R_0 - I = 0, \tag{4}$$

and the distal boundary condition is

$$\boldsymbol{\psi}'_\ell = \mathbf{0}, \tag{5}$$

assuming that when a tube is not physically present at an arc-length, it has infinite torsional stiffness and zero bending stiffness. The inputs are

$$\boldsymbol{\psi}_0 - \boldsymbol{\alpha} = \mathbf{0}, \quad \boldsymbol{\sigma}_0 - \boldsymbol{\gamma} = \mathbf{0}, \tag{6}$$

where $\boldsymbol{\alpha}$ and $\boldsymbol{\gamma}$ are the tubes' *actuator* rotation angles and translations, respectively.

3 State Estimation for Continuum Robots

In this paper, we argue that design, sensing, and planning are fundamentally coupled problems for continuum robots, and that statistical state estimation can be used to understand the interaction between these three problems. The goal of statistical state estimation is to infer a continuum robot's state that is most likely given the prior model (1)–(2) and sensor observations, in the presence of uncertainty in the process, observations, and boundary constraints. We assume that beliefs can be modeled with Gaussian distributions, denoted with $\mathcal{N}(\boldsymbol{x}, \Sigma)$, having mean $\boldsymbol{x}$ and covariance Σ.

Uncertainty in the spatial kinematics of a continuum robot is modeled using a spatial stochastic process given by

$$\boldsymbol{x}'_s = \boldsymbol{f}(\boldsymbol{x}_s, s) + \boldsymbol{q}_s \tag{7}$$

with uncertain constraints and inputs of the form

$$\boldsymbol{b}_0(\boldsymbol{x}_0) = \boldsymbol{w}_0 \qquad \boldsymbol{b}_\ell(\boldsymbol{x}_\ell) = \boldsymbol{w}_\ell \qquad \boldsymbol{u}(\boldsymbol{x}_0) = \boldsymbol{v} \tag{8}$$

where $\boldsymbol{q}_s \sim \mathcal{N}(\mathbf{0}, Q_s)$ represents uncertainty in the process and is independent in arc-length. Uncertainty in the boundary conditions is accounted for by $\boldsymbol{w}_0 \sim \mathcal{N}(\mathbf{0}, W_0)$ and $\boldsymbol{w}_\ell \sim \mathcal{N}(\mathbf{0}, W_\ell)$, which are independent. Uncertainty in the input is represented by $v \sim \mathcal{N}(\mathbf{0}, V)$. Uncertainty in the proximal boundary condition could model uncertainty in the proximal pose of a continuum robot. Uncertainty in the distal boundary condition could model uncertainty in the distal torsional moment for a concentric-tube robot, which is ideally moment-free without applied loading.

We denote a set of time-invariant observation functions $\boldsymbol{h}_{s_1}, \ldots, \boldsymbol{h}_{s_m}$ to be defined at discrete arc-lengths $s_1, \ldots, s_m$ as

$$\boldsymbol{y}_{s_i} = \boldsymbol{h}_{s_i}(\boldsymbol{x}_{s_i}) + z_{s_i}, \tag{9}$$

where i indicates the i^{th} observation and $i = 1, \ldots, m$. Note that we do not assume the measurements are evenly spaced or that the dimension of each measurement is the same (i.e., the measurements can be gathered by a heterogeneous collection of sensors placed arbitrarily along the length of the continuum robot). The observations are subject to additive noise $z_{s_i} \sim N(\mathbf{0}, Z_{s_i})$, and are mutually uncorrelated.

3.1 Spatial Statistical Estimation

Statistical estimation infers the continuum robot's state-curve given the model (7), uncertain inputs and constraints (8), and all noisy sensor observations on the robot's body. Estimation requires that the states be observable from sensor measurements [34]. The best estimate is the smoothed estimate, denoted by $\mathcal{N}(\bar{\boldsymbol{x}}_s, \bar{P}_s)$ with expected value $\bar{\boldsymbol{x}}_s$ and covariance $\bar{P}_s$, and can be approximated in a two-step procedure:

(1) The first step recursively computes the *posterior* estimate at any arc-length s, denoted by $\mathcal{N}(\tilde{\boldsymbol{x}}_s, \tilde{P}_s)$, which incorporates the sensor observations in the arc-length interval $[0, s]$, the proximal boundary constraints $\boldsymbol{b}_0$, and the inputs $\boldsymbol{u}$. The *posterior* estimate is found using an extended Kalman filter that has become ubiquitous throughout robotics [38], and is obtained by continuously propagating a *prior* state estimate, denoted by $\mathcal{N}(\hat{\boldsymbol{x}}_s, \hat{P}_s)$, down the length of the robot's backbone using the extended Kalman-Bucy equations, and updating it at measurement locations s_i to form the *posterior* estimate. The *prior* state estimate is the belief given all sensor measurements obtained in the arc-length interval $[0, s)$. If no sensor measurement is present at s, then the *prior* and *posterior* estimates are the same.

The *prior* estimate is propagated by piecewise-integrating the equations

$$\hat{\boldsymbol{x}}'_s = \boldsymbol{f}(\hat{\boldsymbol{x}}_s, s) \tag{10}$$

$$\hat{P}'_s = F_s \hat{P}_s + \hat{P}_s F_s^{\mathsf{T}} + Q_s \tag{11}$$

from $s = 0$ to ℓ in the intervals $[0, s_1], [s_{i-1}, s_i], \ldots, [s_m, \ell]$. The initial conditions, $\hat{\boldsymbol{x}}_0$ and $\hat{P}_0$, represent prior information that could be found by solving (1)–(2).

The *prior* estimate is updated to form the *posterior* estimate with a Kalman update at every sensor observation at arc-lengths $s_1 \cdots s_m$, of the form

$$\hat{S}_{s_i} = \hat{H}_{s_i} \hat{P}_{s_i} \hat{H}_{s_i}^{\mathsf{T}} + Z_{s_i} \tag{12}$$

$$\hat{K}_{s_i} = \hat{P}_{s_i} \hat{H}_{s_i}^{\mathsf{T}} \hat{S}_{s_i}^{-1} \tag{13}$$

$$\tilde{\boldsymbol{x}}_{s_i} = \hat{\boldsymbol{x}}_{s_i} + \hat{K}_{s_i} \left(\boldsymbol{y}_{s_i} - \boldsymbol{h}_{s_i}(\hat{\boldsymbol{x}}_{s_i})\right) \tag{14}$$

$$\tilde{P}_{s_i} = (I - \hat{K}_{s_i} \hat{H}_{s_i}) \hat{P}_{s_i}. \tag{15}$$

The inputs and boundary conditions (8) can be incorporated using the Kalman update Eqs. (12)–(15). For example, the distal boundary condition can be incorporated as a Kalman update with $\hat{H}_{s_i}$, Z_{s_i}, and $\boldsymbol{y}_{s_i}$ replaced with the matrix $\partial \boldsymbol{b}_\ell / \partial \boldsymbol{x}_\ell$, W_ℓ and $\mathbf{0}$, respectively. In theory, the incorporation of constraints in this way also works if there is no uncertainty in the constraints (i.e., the constraint covariances,

W_0 or W_ℓ, are not invertible). Care must be taken in this case since numerical issues while integrating (11) may result in eigenvalues of the estimate covariance becoming negative, making the covariance no longer positive-definite. In this case, square-root formulations of the filtering equations can be used to increase precision and ensure that the positive semidefinite-ness is preserved [10].

(2) The *posterior* estimate is then used by the second step, which computes the state estimate at arc-length s that incorporates all sensor observations everywhere on the robot, the proximal and distal boundary constraints, and the inputs using an extended form of a Kalman smoother. Smoothing is frequently used as a method for batch post-process analysis, after all data has been gathered, to determine the best state estimates of the past given all gathered knowledge [34].

The smoothed estimate can be computed from the *posterior* estimate by propagating the Rauch-Tung-Striebel (RTS) differential equations

$$\bar{\boldsymbol{x}}_s' = \boldsymbol{f}(\bar{\boldsymbol{x}}_s, s) + Q_s \tilde{P}_s^{-1} (\bar{\boldsymbol{x}}_s - \tilde{\boldsymbol{x}}_s) \tag{16}$$

$$\bar{P}_s' = \left(F_s + Q_s \tilde{P}_s^{-1}\right) \bar{P}_s + \bar{P}_s \left(F_s + Q_s \tilde{P}_s^{-1}\right)^{\mathsf{T}} - Q_s \tag{17}$$

backward from arc-length ℓ to 0, with initial conditions $\bar{\boldsymbol{x}}_\ell = \tilde{\boldsymbol{x}}_\ell$ and $\bar{P}_\ell = \tilde{P}_\ell$.

3.2 Experimental Results

We applied the methods presented herein to a three-tube concentric-tube robot, modeled by Eqs. (3)–(6), and shown in Fig. 2i. The sensing system incorporates

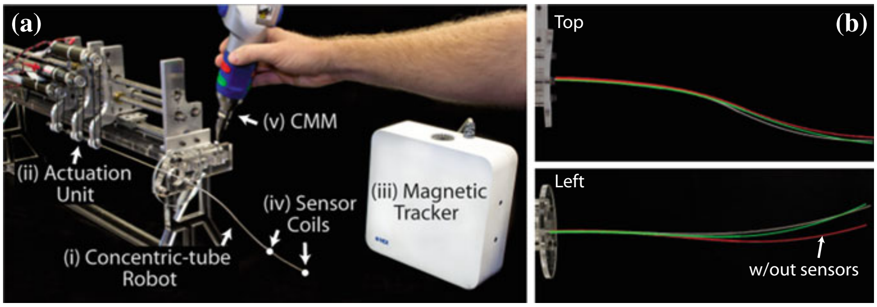


Fig. 2 **a** The experimental setup consists of a three-tube concentric-tube robot (*i*), controlled with a 6-DoF motorized actuation unit (*ii*), and with a electromagnetic sensing-system (*iii*) that tracks two sensor coils (*iv*) embedded in the robot's inner tube. A coordinate measuring machine (CMM) is used to obtain ground-truth measurements of the robot-tip position (*v*). **b** An example configuration of the concentric-tube robot with the shape (position) estimate, that incorporates magnetic position sensors, is shown with the shape estimate without magnetic position-sensors along with the actual shape of the robot. For the configuration shown, the tip-position errors were 7.3 and 16.0 mm for the estimate with and without sensor information, respectively

information from the six encoders of the actuation unit (Fig. 2ii) and an NDI Aurora electromagnetic-tracking system (Fig. 2iii) that measures the position of two sensing-coils placed in the concentric-tube robot as shown in Fig. 2iv. A FARO Gage coordinate measuring machine (Fig. 2v) was used to measure registration transformation between the robot and electromagnetic tracker as well as for obtaining ground-truth tip-position used in the experiments. We found that the six encoder measurements and the measured registration are sufficient to guarantee observability as discussed in Sect. 3.1, under the zero-tip-moment assumption ($\boldsymbol{\psi}''_{\ell} = \mathbf{0}$).

The magnetic-tracker measurements add additional information that improves the state estimate under modeling uncertainty. We experimentally demonstrate their benefit by placing the concentric-tube robot in four configurations and comparing the error in tip-position provided by the smoothed estimate with the magnetic-tracker information to the smoothed estimate without tracker information. The tip-position error was measured using the CMM. On average, the accuracy of the smoothed estimate with and without tracker information was 8.8 and 14.5 mm, respectively. Figure 2b shows the shape estimate with and without the magnetic-tracking sensors for one of the four configurations used in our experiments.

The smoothed estimate for the states of the three-tube concentric-tube robot is implemented with the initial covariance of the tube-twist angles $\boldsymbol{\psi}_0$ set to $0.08I$ rad^2, where I is a 3×3 identity matrix, the initial proximal covariance of the tube-twist-angle rates-of-change $\boldsymbol{\psi}'_0$ are set to $0.01I$ (rad/m)^2, the initial proximal covariance of the backbone position $\boldsymbol{p}_0$ are set to $5 \times 10^{-5} I$ m^2. The covariances of the initial proximal backbone orientation (represented as a quaternion) and the concentric-tube translations $\boldsymbol{\sigma}_0$ are assumed to be zero. These initial covariances are found using estimates of calibration accuracy and are used as the initial values for the state-covariance in the computation of the posterior state-estimate obtained by integrating the Kalman-Bucy filter equations, (10) and(11). The position covariance of the magnetic tracker is experimentally found to be $1 \times 10^{-6} I$ m^2, which is used in the Kalman update step (12)–(15) for the posterior state-estimate.

4 Connections Between Estimation and Design, Sensing, and Planning

4.1 Connections Between Mechanical Design and Estimation

When the application requires the continuum robot to be sensed, the mechanical design of the robot plays an important role in estimation through the kinematic model (1)–(2), which is used in the Kalman-Bucy filter to condition the uncertain sensor observations. The mechanical design affects the quality of the state estimate (i.e., the covariance), through the linearized-kinematics state matrix F_s, which appears in differential equations (11) and (17) that govern the propagation of the prior and smoothed covariances along the length of the continuum robot. The physicall

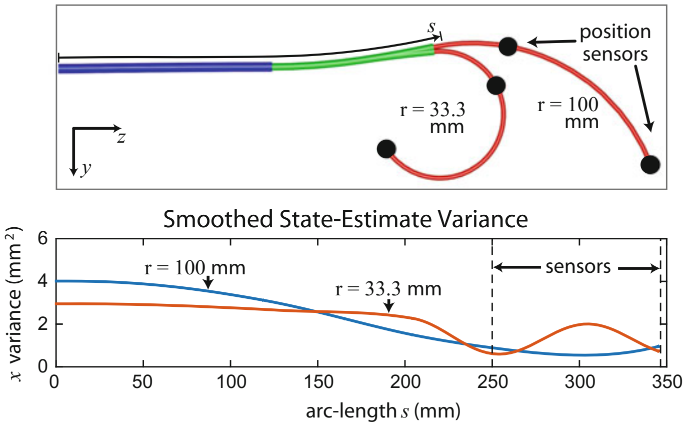


Fig. 3 The effect of changing a concentric-tube robot's inner-tube curvature on the smoothed state-estimate position variance in the x direction (out of the page) is shown (the variance in the y and z are largely unchanged). The smoothed estimate is found using the methods in Sect. 3.1 with two backbone-position sensors arranged on the robot as depicted. The covariances used by the smoothed estimator are for the physical robot reported in Sect. 3.2

properties of the robot that affect the matrix F_s could be, in the case of a tendon-actuated robot with a backbone, the robot's backbone stiffness or the distance of routed tendons to the backbone. In the case of a concentric-tube robot, physical properties can also include tube precurvatures and tube stiffnesses. The physica properties could vary along the length of the robot. For example, one or more flexure-hinges along the backbone (e.g., [50]) would locally decrease its stiffness.

As an example of how a continuum robot's mechanical design could affect state estimation, we study how changing the curvature of a concentric-tube robot's innermost tube changes the variance of the smoothed position estimate. We use two position sensors placed as shown in Fig. 3, with the methods of Sect. 3.1, using the robot and covariance parameters reported in Sect. 3.2. We now explore the effect on the covariance of changing the inner-tube's radius of curvature from 100 to 33.3 mm. The x-variance (out of the page) of the smoothed position estimate for each radius is shown in Fig. 3 (the variance in the y and z directions is largely unchanged), plotted as a function of backbone arc-length s. As indicated, the x variance increases between the two sensors (i.e., between arc-lengths at 250 and 350 mm) as the curvature increases. In this region, the robot's backbone position is most sensitive to rotation of the inner tube, which causes uncertainty in the inner-tube's rotation to manifest itself more in the backbone uncertainty, particularly in the x direction.

A nonintuitive effect occurs for proximal arc-lengths between 0 mm and approximately 150 mm. In this region, higher inner-tube curvature tends to reduce the x backbone-position variance. This could be an effect resulting from the geometry of the concentric-tube robot's proximal end and the sensor placements on the robot's backbone. Naturally, it would be difficult to maneuver a robot through a narrow passageway in this configuration. In order to make the most of this effect, the robot's curvature should be designed in coordination with a motion planner in order to account for constraints imposed by the robot's environment.

4.2 Connections Between Sensor Selection, Sensor Placement, and Estimation

The question of what sensors should be chosen and their placement, is a frequent concern of continuum robot designers. Often there are space limitations due to the requirement that the diameter be small (e.g., to navigate through blood vessels) and there is little room left over once actuators have been installed. Therefore, it is desirable to obtain as much state information as possible, with as few sensors as possible.

The kinematic structure of the continuum robot, the robot's configuration, and the states that the sensors observe all contribute to the amount of state information obtained by the sensors. An example is shown for a concentric-tube robot in Fig. 4, where the smoothed state covariance in the x direction is shown for two arrangements of sensors. The covariance in the other two directions are largely unchanged. The covariances used by the smoothed estimator to obtain the results of Fig. 4 are for the physical robot reported in Sect. 3.2. Clearly, sensor placement has a large influence on the covariance of the smoothed backbone-position estimate. If an application requires the position estimate to be more accurate in the arc-length region $(150, 300)$ mm, then placement "1" would be preferred over placement "2".

The variation in the smoothed covariance with variation in sensor placement can be exploited by the designer to best meet the needs of the application. The first task

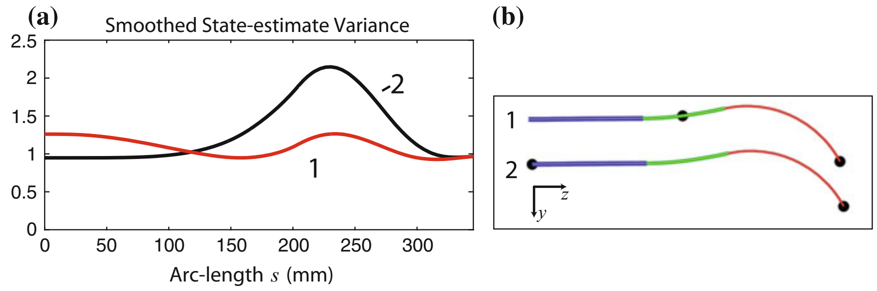


Fig. 4 The effect of changing sensor-observation placement is illustrated in this example where the smoothed state-estimate covariance in the x direction is shown (**a**) for two sensor arrangements (**b**). In *1* the one sensor is placed in the middle of the robot and one at the distal end, while in *2* one sensor is placed at the most proximal and most distal ends of the robot. The position-estimate covariance in the other two directions is largely unchanged. The covariances used by the smoothed estimator are for the physical robot reported in Sect. 3.2

is to select the appropriate sensors and sensor locations to achieve application specifications when states are estimated as described in Sect. 3.1. For many applications, partial state information may be sufficient. In others, accuracy requirements may vary by direction or arc-length. Based on such specifications, the methods in this paper enable one to determine how many sensors are required (e.g., to ensure observability described in Sect. 3.1).

One way to encode the requirements of the application is through a covariance-based metric, over which optimization in the sensor positions can be performed. If there are several arc-length points-of-interest $\gamma_1, \ldots, \gamma_j$ where the uncertainty of a continuum robot's state estimate is critical in specific directions (e.g., to ensure that the position estimate in the direction of nearest obstacles is highly certain), then an example of a metric on the smoothed covariance at the points-of-interest could be

$$\mathscr{U}(\bar{P}_{\gamma_1}, \ldots, \bar{P}_{\gamma_k}) = \max_{i=1\ldots j} \operatorname{tr}(E_i^\top \bar{P}_{\gamma_i} E_i) \tag{18}$$

where tr denotes the matrix trace, and E_i is a positive-semidefinite matrix used to select the critical directions of covariance, to selectively assign priority to arc-length the points-of-interest on the robot, or enforce unit-consistency. This metric minimizes the worst-case average estimate-variance at the arc-length points-of-interest.

4.3 Connections Between Motion Planning and Estimation

A major benefit of continuum robots is that they have the potential to snake through constrained spaces. The basic motion planning problem is to find a time-ordered sequence of continuum-robot inputs $\boldsymbol{u}_1(\boldsymbol{x}_0), \ldots, \boldsymbol{u}_k(\boldsymbol{x}_0)$, for time indices $1 \cdots k$, that guide the robot through the constrained space without colliding with obstacles in the environment. Motion planning is particularly relevant to surgical applications of continuum robots, where physicians desire to reach a surgical target while avoiding anatomical obstacles (e.g., bones, blood vessels, and sensitive organs) or danger zones established by the physician that correspond to high risk areas.

To avoid obstacles, motion planners must have knowledge over time of the robot's state, which for continuum robots includes the robot's shape (e.g., position-curve parameterized by arc-length s). In practice, there is often substantial uncertainty in the estimate of the robot's shape, and the motion planner must account for the shape uncertainty to guarantee that obstacles will be avoided with a desired level of confidence. To address this challenge, rather than following the traditional workflow of computing a motion plan assuming perfect state estimation and then relying on a real-time controller to account for uncertainty, recent work has investigated integrating motion planning with control to compute motion policies, which are parameterized by time and sensor measurements. Combining motion planning, control, and state estimation into a single problem can result in higher quality plans. General motion planning methods linking these problems have been developed for robots for which

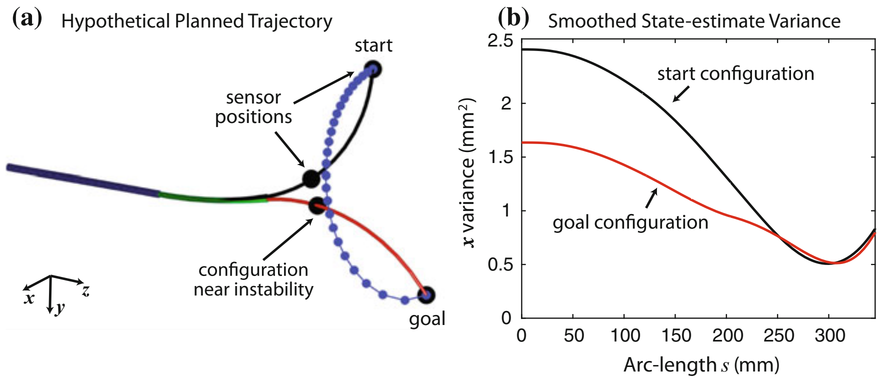


Fig. 5 The effect of changing the input $\boldsymbol{u}(\boldsymbol{x}_0)$ to follow a hypothetical trajectory from a start to goal configuration that could have been generated as an obstacle-avoiding trajectory by a motion planner. The variance in the x-direction of the concentric-tube robot's position-covariance is shown at both configurations. (The position-variance in the other directions are largely unchanged.) The covariances used by the smoothed estimator are for the physical robot reported in Sect. 3.2

state uncertainty can be modeled using Gaussian distributions in the state space. These belief-space planners, e.g., [6, 11, 20, 25–28, 45, 46], often use a sampling-based or optimization-based motion planner integrated with an automatically-tuned linear controller, and variants have been specifically applied to medical robots such as steerable needles and concentric-tube robots based on simplified uncertainty models [35, 36, 44]. Better models of the belief space of continuum robots (i.e., the space of distributions of state estimates given specific sensor measurements) could enable the computation of more effective motion policies.

For concentric-tube robots, some configurations naturally result in lower smoothed covariances than others. To illustrate the information available to a motion planner, Fig. 5 shows the smoothed state-estimate variance in the x direction as the physical concentric-tube robot from Sect. 3.2 follows a hypothetical trajectory, where the inner tube is rotated from the start-configuration to the end-configuration as shown in Fig. 5a. As the concentric-tube robots nears the goal-configuration, the smoothed variance of the robot's shape estimate in the x-direction decreases (the estimate improves) by 35% at the proximal end of the robot with little change in the distal end. The covariances used by the smoothed estimator are for the physical robot reported in Sect. 3.2. For motion planners to fully exploit the natural reduction in smoothed covariance, the temporal state transition and uncertainty in the transition must be understood and incorporated into a full spatiotemporal estimate. Creating efficient belief space planners for continuum robots that are based on accurate kinematic and uncertainty models is a significant research challenge.

Note that the covariance of the proximal position estimate represents the position uncertainty of the robot's pose in space. For some surgical applications, the robot's pose is could be intraoperatively registered to a patient's anatomy. The initial position covariance in the x direction for the experimental system in Sect. 3.2 (with which

Fig. 5 was computed) was estimated to be 50 mm^2. The process of Kalman smoothing can be used to improve the registration using the sensor observations taken along the robot's backbone. This is shown in Fig. 5, where position observations generated by the the electromagnetic tracker improve the position covariance of the registration in the x direction to less than 3.0 mm^2. In the case of the robot in the "goal" configuration, the covariance is improved to 1.6 mm^2. It may be beneficial for a motion planner to direct the robot near configurations where the initial registration can be improved before starting delicate surgical procedures.

When placed in certain configurations, some concentric-tube-robot designs exhibit elastic instability, which is observed as a windup, and sudden release of torsion. The release of torsion results in fast, uncontrolled motion of the robot's distal end. Passing through an elastic instability could be extremely harmful for sensitive surgical manipulation tasks. As a result, it is generally assumed that concentric-tube robots should be operated in configurations that are far from instability [3]. One nonintuitive result of Fig. 5, however, is that configurations near instabilities may be more "information rich" than those far away. This is indicated by the fact that the "goal" configuration, which is near what we understand to be the most likely configuration where instability occurs, has lower proximal position covariance than the "start" configuration, which is near what we know to be a configuration far from instability [14]. This indicates that configurations near instability may be useful for mitigating increases in the concentric-tube robot's state-estimate uncertainty as it follows a trajectory, and we expect that a motion planner that explicitly considers uncertainty could favor configurations near (but not at) instability at times for this reason.

5 Conclusion

In this paper, we used statistical state estimation as a tool to study the problems of designing a continuum robot's geometry, selecting and placing sensors for detecting the continuum robot, and using motion planning to direct the trajectory of a continuum robot to accomplish a task. These three problems are fundamentally coupled and their interaction can be studied using the covariance produced by statistical state estimation. The future of continuum robots lies in the simultaneous solution to the continuum-robot design, sensing, and planning problems in order to produce continuum-robot systems that make the most of their geometry and available information to satisfy the needs of the most demanding applications.

References

1. Ayvali, E., Liang, C.-P., Mingyen, H., Chen, Y., Desai, J.P.: Towards a discretely actuated steerable cannula for diagnostic and therapeutic procedures. Int. J. Rob. Res. **31**(5), 588–603 (2012)
2. Baykal, C., Torres, L.G., Alterovitz, R.: Optimizing design parameters for sets of concentric tube robots using sampling-based motion planning. In: Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems, pp. 4381–4387 (2015)
3. Bergeles, C., Dupont, P.E.: Planning stable paths for concentric tube robots. In: Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems, pp. 3077–3082 (2013)
4. Bergeles, C., Gosline, A.H., Vasilyev, N.V., Codd, P.J., del Nido, P.J., Dupont, P.E.: Concentric tube robot design and optimization based on task and anatomical constraints. IEEE Trans. Robot. **31**(1), 67–84 (2015)
5. Borum, A., Matthews, D., Bretl, T.: State estimation and tracking of deforming planar elastic rods. In: Proceedings of IEEE International Conference on Robotics and Automation, pp. 4127–4132 (2014)
6. Bry, A., Roy, N.: Rapidly-exploring random belief trees for motion planning under uncertainty. In: Proceedings of IEEE International Conference on Robotics and Automation, pp. 723–730 (2011)
7. Bryson, C.E., Rucker, D.C.: Toward parallel continuum manipulators. In: Proceedings of IEEE International Conference on Robotics and Automation, pp. 778–785 (2014)
8. Burgner, J., Gilbert, H.B., Webster III, R.J.: On the computational design of concentric tube robots: incorporating volume-based objectives. In: Proceedings - IEEE International Conference on Robotics and Automation, pp. 1193–1198 (2013)
9. Burgner-Kahrs, J., Rucker, D.C., Choset, H.: Continuum robots for medical applications: a survey. IEEE Trans. Robot., (2015, in press)
10. Crassidis, J.L., Junkins, J.L.: Optimal Estimation of Dynamic Systems, 2nd edn. CRC Press, Boca Raton (2011)
11. Du, Toit, Noel, E., Burdick, J.W.: Robot motion planning in dynamic, uncertain environments. IEEE Trans. Robot. **28**(1), 101–115 (2012)
12. Dupont, P.E., Lock, J., Itkowitz, B., Butler, E.: Design and control of concentric-tube robots. IEEE Trans. Robot. **26**(2), 209–225 (2010)
13. Gilbert, H.B., Rucker, D.C., Webster III, R.J.: Concentric tube robots: state of the art and future directions. Proc. Int. Symp. Robot. Res., (2013, in press)
14. Gilbert, H.B., Hendrick, R.J., Webster, III., R.J.: Elastic Stability of Concentric Tube Robots. IEEE Trans. Rob. (2015)
15. Hannan, M.W., Walker, I.D.: Real-time shape estimation for continuum robots using vision. Robotica **23**(5), 645–651 (2005)
16. Hirose, S.: Biologically Inspired Robots, Snake-Like Locomotors and Manipulators. Oxford University Press, Oxcord (1993)
17. Jeon, J., Yi, B.-J.: Shape prediction algorithm for flexible endoscope. In: Proceedings of IEEE International Conference on Robotics and Automation, pp. 2856–2861 (2014)
18. Jones, B.A., Walker, I.D.: Kinematics for multisection continuum robots. IEEE Trans. Robot. **22**(1), 43–55 (2006)
19. Koolwal, A.B., Barbagli, F., Carlson, C., Liang, D.: An ultrasound-based localization algorithm for catheter ablation guidance in the left atrium. Int. J. Rob. Res. **29**(6), 643–665 (2010)
20. Lee, A., Duan, Y., Patil, S., Schulman, J., McCarthy, Z., van den Berg, J., Goldberg, K., Abbeel, P.: Sigma hulls for gaussian belief space planning for imprecise articulated robots amid obstacles. In: Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems, pp. 5660–5667 (2013)
21. Lobaton, E.J., Fu, J., Torres, L.G., Alterovitz, R.: Continuous shape estimation of continuum robots using X-ray images. In: Proceedings of IEEE International Conference on Robotics and Automation, pp. 725–732 (2013)

22. Lyons, L.A., Webster III, R.J., Alterovitz, R.: Motion planning for active cannulas. In: Proceedings of IEEE/RSJ International Conference on Intelligent Robots and System, pp. 801–806 (2009)
23. Novotny, P.M., Stoll, J.A., Dupont, P.E., Howe, R.D.: Real-time visual servoing of a robot using three-dimensional ultrasound. In: Proceedings of IEEE International Conference on Robotics and Automation, pp. 2655–2660 (2007)
24. Odhner, L.U., Jentoft, L.P., Claffee, M.R., Corson, N., Tenzer, Y., Ma, R.R., Buehler, M., Kohout, R., Howe, R.D., Dollar, A.M.: A compliant, underactuated hand for robust manipulation. Int. J. Rob. Res. **33**(5), 736–752 (2014)
25. Patil, S., van den Berg, J., Alterovitz, R.: Motion planning under uncertainty in highly deformable environments. In: Proceedings of Robotics: Science and Systems (2011)
26. Patil, S., van den Berg, J., Alterovitz, R.: Estimating probability of collision for safe motion planning under gaussian motion and sensing uncertainty. In: Proceedings of IEEE International Conference on Robotics and Automation, pp. 3238–3244 (2012)
27. Platt Jr., R., Tedrake, R., Kaelbling, L., Lozano-Perez, T.: Belief space planning assuming maximum likelihood observations. In: Proceedings of Robotics: Science and Systems (2010)
28. Prentice, S., Roy, N.: The belief roadmap: efficient planning in belief space by factoring the covariance. Int. J. Rob. Res. **28**(11), 1448–1465 (2009)
29. Roesthuis, R.J., Kemp, M., van den Dobbelsteen, J.J., Misra, S.: Three-dimensional needle shape reconstruction using an array of fiber bragg grating sensors. IEEE/ASME Trans. Mech. **19**(4), 1115–1126 (2014)
30. Rucker, D.C., Webster, R.J.: Statics and dynamics of continuum robots with general tendon routing and external loading. IEEE Trans. Robot. **27**(6), 1033–1044 (2011)
31. Rucker, D.C., Jones, B.A., Webster III, R.J.: A geometrically exact model for externally loaded concentric-tube continuum robots. IEEE Trans. Robot. **26**(5), 769–780 (2010)
32. Ryu, S.C., Dupont, P.E.: FBG-based shape sensing tubes for continuum robots. In: Proceedings IEEE International Conference on Robotics and Automation, pp. 3531–3537 (2014)
33. Ryu, S.C., Quek, Z.F., Koh, J.-S., Renaud, P., Black, R.J., Moslehi, B., Daniel, B.L., Cho, K.-J., Cutkosky, M.R.: Design of an optically controlled mr-compatible active needle. IEEE Trans. Robot. **31**(1), 1–11 (2015)
34. Simon, D.: Optimal State Estimation: Kalman, H_∞, and Nonlinear Approaches. Wiley, New Jersey (2006)
35. Sun, W., Alterovitz, R.: Motion planning under uncertainty for medical needle steering using optimization in belief space. In: Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems, pp. 1775–1781 (2014)
36. Sun, W., Torres, L.G., van den Berg, J., Alterovitz, R.: Safe motion planning for imprecise robotic manipulators by minimizing probability of collision. In: Proceedings of International Symposium Robotics Research (2013)
37. Swaney, P.J., Burgner, J., Pheiffer, T.S., Rucker, D.C., Gilbert, H.B., Ondrake, J.E., Simpson, A.L., Burdette, E.Clif, Miga, M.I., Webster III, R.J.: Tracked 3D ultrasound targeting with an active cannula. In: Proceedings of the Society of Photo-Optical Instrumentation Engineers, pp. 83160R–83160R–9 (2012)
38. Thrun, S., Burgard, W., Fox, D.: Prob. Robot. MIT Press, Cambridge (2006)
39. Torres, L.G., Alterovitz, R.: Motion planning for concentric tube robots using mechanics-based models. In: Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems, pp. 5153–5159 (2011)
40. Torres, L.G., Webster III, R.J., Alterovitz, R.: Task-oriented design of concentric tube robots using mechanics-based models. In: Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems, pp. 4449–4455 (2012)
41. Torres, L.G., Baykal, C., Alterovitz, R.: Interactive-rate motion planning for concentric tube robots. In: Proceedings of IEEE International Conference on Robotics and Automation, pp. 1915–1921 (2014)
42. Torres, L.G., Kuntz, A., Gilbert, H.B., Swaney, P.J., Hendrick, R.J., Webster III, R.J., Alterovitz, R.: A motion planning approach to automatic obstacle avoidance during concentric tube robot

teleoperation. In: Proceedings of the IEEE International Conference on Robotics and Automation, pp. 2361–2367 (2015)
43. Trovato, K., Popovic, A.: Collision-free 6D non-holonomic planning for nested cannulas. In: Proceedings of SPIE Medical Imaging, vol. 7261 (2009)
44. van den Berg, J., Patil, S., Alterovitz, R., Abbeel, P., Goldberg, K.: LQG-based planning, sensing, and control of steerable needles. In: Hsu, D. et al. (eds), Algorithmic Foundations of Robotics IX. Springer Tracts in Advanced Robotics, vol. 68, pp. 373–389. Springer, Heidelberg (2010)
45. van den Berg, J., Abbeel, P., Goldberg, K.: LQG-MP: optimized path planning for robots with motion uncertainty and imperfect state information. Int. J. Rob. Res. **30**(7), 895–913 (2011)
46. van den Berg, J., Patil, S., Alterovitz, R.: Motion planning under uncertainty using iterative local optimization in belief space. Int. J. Rob. Res. **31**(11), 1263–1278 (2012)
47. Webster III, R.J., Jones, B.A.: Design and kinematic modeling of constant curvature continuum robots: a review. Int. J. Rob. Res. **29**(13), 1661–1683 (2010)
48. Xu, K., Simaan, N.: Analytic formulation for kinematics, statics, and shape restoration of multibackbone continuum robots via elliptic integrals. ASME J. Mech. Robot. **2**(1), 011006 (2009)
49. Xu, R., Patel, R.V.: A fast torsionally compliant kinematic model of concentric-tube robots. In: Proceedings of International Conference IEEE Engineering in Medicine and Biology Society, pp. 904–907 (2012)
50. York, P., Swaney, P.J., Gilbert, H.B, Webster III, R.J.: A wrist for needle-sized surgical robots. In: Proceedings of IEEE International Conference on Robotics and Automation, pp. 1776–1781 (2015)

Part III
Hands and Haptics

Session Summary

As the new wave of humanoid robots is pushing the boundaries of what can be achieved outside of traditional, fully structured environments, the importance of manipulation skills is growing increasingly clear. In applications from disaster response to e-commerce order picking and in-home assistance, these robots must be able to effect change on their surrounding environment, interacting with and manipulating a wide range of objects. However, the development of such skills comprises numerous aspects, from mechanism design to control, planning, and teleoperation. Beyond autonomous robots, advances in manipulation and haptics can also translate into improved assistive or rehabilitative devices intended to directly interact with the human body. This can include orthoses, prostheses, and wearable robots. This session presented new advances in many of these areas.

A key area of interest is the ability to move beyond simple scenes with isolated objects to grasping and manipulating in real clutter. The work of ten Pas and Platt presents a new approach to grasp planning in cluttered scenes, without requiring existing models of the objects to be manipulated. Dai et al. also present an efficient method for synthesizing and optimizing grasps using the sequential semidefinite programming. Addressing the problems or variance and variability in grasping, Jentoft et al. introduce a framework for the grasping problem that considers grasp robustness but also the uncertainty arising from many components of the system; Quispe et al. introduce a novel taxonomy of benchmark tasks for manipulation, drawing on the rich related literature in physical therapy and rehabilitation.

In the area of assistive robotic manipulators, Ying et al. present a new method for shared-control online grasp planning where human input is provided directly through a brain–computer interface. Tokatli and Patoglu discuss haptic rendering from a mathematical perspective, using fractional order models to render impedances beyond the reach of traditional models. Last but not least, two papers discuss robotic hardware and novel control methods. Wu and Carricato present the design of a serial-parallel 3-degree-of-freedom robotic wrist with a singularity-free 180-degree pointing cone; Asano et al. introduce a balancing controller for a humanoid robot combining the ZMP approach with redundant actuation; Salvietti et al. introduce a novel wearable extra robot finger for rehabilitation of the paretic limb.

Overall, the presented papers range from design and control to grasp planning, taxonomies, and encompassing frameworks for manipulation. These are all advances that aim to push complete manipulation systems closer to the goal of achieving versatility and robustness in increasingly more difficult domains.

Synthesis and Optimization of Force Closure Grasps via Sequential Semidefinite Programming

Hongkai Dai, Anirudha Majumdar and Russ Tedrake

1 Introduction

Force closure, which measures the ability of a grasp to resist wrench disturbances, is an important property in grasping and has an extensive literature [17, 18]. A commonly observed fact is that synthesis of force closure grasps is a non-convex optimization problem, mostly due to the fact that computing the torque on an object involves a bilinear product between contact locations and contact forces. As a result, most approaches resort to gradient-based non-convex nonlinear optimization to synthesize a force closure grasp [5]. On the other hand, when fixing the contact locations, checking if the given contact achieve force closure becomes a convex optimization problem over contact forces only [3, 10]. Moreover, several grasp metrics have been introduced to measure the quality of a force closure grasp. These involve computing the smallest wrench that the grasp cannot resist with bounded contact forces [9, 12, 15]. Liu et al. optimized the contact locations based on such a metric with nonlinear optimization [13].

In this paper we exploit the observation that although the force-closure grasp synthesis problem involves non-convex constraints, the bilinear structure of the constraints makes it special. In particular we pose the problem of finding (and optimizing)

This work is supported by David S.Y. and Harold Wong Fellowship, and ONR MURI grant N00014-09-1-1051.

H. Dai (✉) · A. Majumdar · R. Tedrake
Computer Science and Artificial Intelligence Lab, MIT, Cambridge, MA, USA
e-mail: daih@csail.mit.edu

A. Majumdar
e-mail: anirudha@csail.mit.edu

R. Tedrake
e-mail: russt@csail.mit.edu

A. Bicchi and W. Burgard (eds.), *Robotics Research*, Springer Proceedings in Advanced Robotics 2, DOI 10.1007/978-3-319-51532-8_18

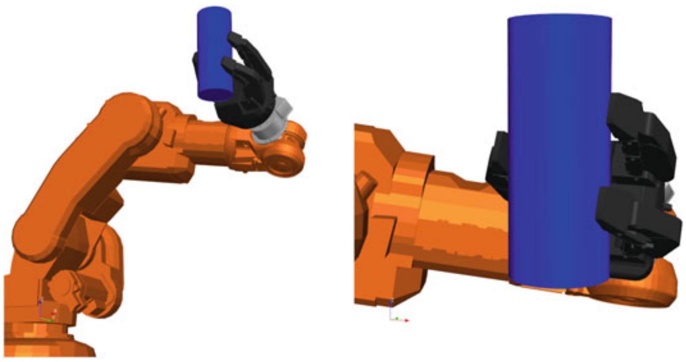


Fig. 1 Optimized force closure grasp of a 15 joints robot on a cylinder, from two perspectives

a force closure grasp as a *Bilinear Matrix Inequality* (BMI). There exist powerful tools to solve BMIs via sequences of semidefinite programming problems (SDP), which is a special form of convex optimization.

Besides satisfying the force closure constraint, the contact locations should also be reachable by the hand, subject to robot kinematics constraints. Traditionally finding robot posture is solved by the Jacobian transpose method [4] or nonlinear optimization [7] on robot minimal coordinates, which involve non-polynomial (e.g., trigonometric) functions. Recently Rosales et al. searched for hand posture by solving linear and bilinear equations of robot maximal coordinates, through an iterative linear optimization scheme [20]. We will adopt the similar idea here to formulate the inverse kinematics problem as BMIs, and solve them through sequential semidefinite programming (Fig. 1).

Sequential semidefinite programming is a common technique in solving bilinear matrix inequalities (BMI) [8, 11]. It relaxes the original non-convex problem to a convex SDP, and in each iteration solves a convex relaxation until convergence. The advantage of this approach over gradient based nonlinear-optimization include

1. The ability to incorporate non-smooth *positive semidefinite* (psd) constraints, which appear frequently in grasp planning. The gradient-based nonlinear optimization cannot handle such constraints gracefully due to non-smoothness.
2. Proof of infeasibility. If the relaxed convex problem is infeasible, then the original non-convex problem is also infeasible. Nonlinear optimization cannot guarantee that a problem is globally infeasible, when locally it fails to find a solution.

We will introduce the background on solving BMI with sequential SDP in Sect. 2, and elaborate how this technique can be applied to synthesis and optimization of force closure grasps in Sect. 3. Our results are shown in Sect. 4.

2 Background

As we will see in Sect. 3, finding force closure grasps for a broad class of object geometries can be posed as a bilinear matrix inequality (BMI), which is a particular kind of optimization problem. In this section we provide an introduction to BMIs along with methods to find feasible solutions to them.

2.1 *Bilinear Matrix Inequalities*

Bilinear matrix inequalities (BMIs) are problems of the following form:

$$\text{Find} \quad x \in \mathbb{R}^n \tag{1}$$

$$\text{s.t.} \quad F_0 + \sum_{i=1}^{N} x_i F_i + \sum_{i=1}^{N}\sum_{j=1}^{N} x_i x_j F_{ij} \succeq 0, \tag{2}$$

where F_0, F_i, F_{ij} are constant $m \times m$ symmetric matrices. $\succeq 0$ means the matrix on the left hand-side is positive semidefinite (psd), i.e. all the eigenvalues are non-negative; the special case is when the matrix is just a scalar, then $\succeq 0$ is the same as ≥ 0. We also note that BMIs include constraints that are both bilinear ($i \neq j$) as well as quadratic ($i = j$).

2.2 *Finding Feasible Solutions to BMIs*

While it is well known that BMIs are NP-hard in general [11], there exist very good heuristic methods based on semidefinite programming (SDP) for solving them. Here we review the method presented in [11] for finding feasible solutions to BMIs.

The first step is to write the BMI (1) as a rank-constrained *Linear Matrix Inequality* (LMI) with an additional variable $X \in \mathbb{R}^{N\times N}$:

$$\text{Find:} \quad x \in \mathbb{R}^N, X \in \mathbb{R}^{N\times N} \tag{3}$$

$$\text{s.t.} \quad F_0 + \sum_{i=1}^{N} x_i F_i + \sum_{i=1}^{N}\sum_{j=1}^{N} X_{ij} F_{ij} \succeq 0, \tag{4}$$

$$M := \begin{bmatrix} X & x \\ x^T & 1 \end{bmatrix} \succeq 0, \tag{5}$$

$$\text{rank}(M) = 1. \tag{6}$$

Here, each occurrence of bilinear terms $x_i x_j$ in (1) has been replaced by the (i, j) element of the decision matrix X. Constraints (5) and (6) have been introduced to ensure that $X = xx^T$, resulting in the problems (3) and (1) having equivalent constraints. We note that without the rank constraint (6), problem (3) is a semidefinite program, which is a particular kind of *convex* optimization problem and can be solved efficiently (e.g., using interior point methods) [2].

The key idea in [11] is to drop the rank constraint in (3) and solve a sequence of SDPs that attempt to minimize the rank of M, as shown in Algorithm 1.

Algorithm 1 Finding feasible solutions to BMIs

```
Minimize trace(X) subject to constraints (4) and (5). If problem is infeasible, then problem (1) is
infeasible. If problem is feasible, initialize $x^{(0)}$ and $X^{(0)}$ with the solution. Initialize $k = 1$.
while ¬converged do
  1. Minimize trace($X^{(k)}$) - $2x^{(k-1)T}x^{(k)}$ subject to the constraints (4) and (5).
  2. Set $k \leftarrow k + 1$
end while
```

Note that the first step in Algorithm 1 is the standard trace heuristic for minimizing the rank of a positive semidefinite matrix [2, 8]. The justification for the proceeding steps in the algorithm is based on the observation that the constraint (5) implies (by the Schur complement lemma) that $X \succeq 0$ and $X - xx^T \succeq 0$. This in turn implies that $\text{trace}(X) - x^T x \geq 0$ with equality holding if and only if $X = xx^T$ (i.e. when we have a feasible solution to (1)). Thus, Algorithm 1 proceeds by linearizing the function $\text{trace}(X) - x^T x$ and minimizing this linearization at every iteration. A termination criterion for Algorithm 1 is provided by the following Lemma in [11].

Lemma 1 ([11]) *The following sequence is bounded below by 0 and non-increasing for $k = 1, 2, \ldots$:*

$$t_k := trace(X^{(k)}) - 2x^{(k-1)T}x^{(k)} + x^{(k-1)T}x^{(k-1)}.$$

Hence, this sequence converges to a value $t_{opt} \geq 0$. Equality holds if and only if $X^{(k)} = x^{(k)}x^{(k)T}$ as $k \to \infty$.

Lemma 1 provides us with a convergence criterion for Algorithm 1. Assuming that the first step in Algorithm 1 is feasible (if this is not the case, the original BMI is infeasible), then convergence of the value of t_k to 0 implies that we have found a feasible solution to the BMI. In the case where t_{opt} is not 0, nothing can be inferred.

2.3 Implementation Details

An important detail in implementing Algorithm 1 is that the SDP constraint (5) can be quite large if one has many decision variables x. However, it is typically the case that a large number of variables do not multiply with each other as bilinear products. Formally, consider a graph whose vertices are the variables in x. Two vertices are connected by an edge if the corresponding variables appear in a bilinear product in some constraint. Then we can partition the variables x into subsets $x_{I_1}, x_{I_2}, \dots, x_{I_k}, \dots, x_{I_K}$ corresponding to the connected components of the graph. We can then replace the constraints (5) and (6) by the following constraints:

$$M_k := \begin{bmatrix} X_{I_k,I_k} & x_{I_k} \\ x_{I_k}^T & 1 \end{bmatrix} \succeq 0, \quad \text{rank}(M_k) = 1,\ \forall k = 1, \dots, K. \tag{7}$$

The cost function in Algorithm 1 is then replaced by the sum of the traces of the matrices X_{I_k,I_k}. While we end up with more psd constraints in general, each constraint involves a smaller matrix. Since SDP solve times typically scale poorly with the size of the largest psd constraint, we observe large computational gains in practice.

Another important implementation detail is to employ a randomization step in Algorithm 1, as described in [11]. In each iteration k of the algorithm, we sample a point $x_{rand}^{(k)}$ from the Gaussian distribution with mean $x^{(k)}$ and covariance $X^{(k)} - x^{(k)}x^{(k)T}$, where $(x^{(k)}, X^{(k)})$ is a solution to the SDP at the k-th iteration, and use cost function $\text{trace}(X^{(k+1)}) - 2x_{rand}^{(k)T}x^{(k+1)}$ in $k+1^{th}$ iteration. In practice, the randomization step prevents the algorithm from getting stuck in local minima.

Finally, we note that while we have restricted ourselves so far to *feasibility* problems, it is also possible to optimize cost functions subject to BMI constraints. We will discuss this in the context of optimizing grasp metrics in Sect. 3.3.

3 Approach

3.1 Force Closure

The force closure property for n grasp points $x_i \in \mathbb{R}^3, i = 1, \dots, n$, is achieved when these grasp points can resist arbitrary external wrenches with contact forces f_i at point x_i lying within the friction cone. Mathematically, force closure is formulated as the existence of x_i and f_i satisfying the following constraints:

$$GG' \succeq \epsilon I_{6\times6} \tag{8a}$$
$$Gf = 0 \tag{8b}$$
$$f_i \in \text{int}(\mathcal{FC}_i) \tag{8c}$$
$$x_i \in \mathcal{S}_i \tag{8d}$$

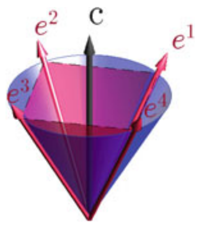


Fig. 2 A nonlinear friction cone (*blue*), and the 4-edge linearized friction cone (*red*). The *red arrows* $e^1, \ldots, e^4$ are the edges of the linearized friction cone. The axis c is along the direction of the normal force, pointing outward from the contact surface

where

$$G = \begin{bmatrix} I_{3\times3} & I_{3\times3} & \ldots & I_{3\times3} \\ \lfloor x_1 \rfloor_\times & \lfloor x_2 \rfloor_\times & \ldots & \lfloor x_n \rfloor_\times \end{bmatrix}, \ \lfloor x_i \rfloor_\times = \begin{bmatrix} 0 & -x_i^{(3)} & x_i^{(2)} \\ x_i^{(3)} & 0 & -x_i^{(1)} \\ -x_i^{(2)} & x_i^{(1)} & 0 \end{bmatrix} \in \mathbb{R}^{3\times3} \tag{9}$$

$\lfloor x_i \rfloor_\times$ is the skew-symmetric matrix representing the cross product $\lfloor x_i \rfloor_\times f_i = x_i \times f_i$, ϵ is a small given positive scalar, constraint (8a) is the same as G being full rank; $f = [f_1^T \ f_2^T \ \ldots f_n^T]^T \in \mathbb{R}^{3n}$; $\text{int}(\mathcal{FC}_i)$ is the interior of the friction cone $\mathcal{FC}_i$ at grasp point x_i, and $\mathcal{S}_i$ is the admissible contact region of grasp point x_i (for example, the surface of the object being grasped).

We note that condition (8a) is quadratic on x_i, and (8b) is bilinear on x_i and f_i. Unlike some existing approaches that fix the contact points x_i and search only contact force f_i through convex optimization, we can search both x_i and f_i simultaneously by solving these BMIs through sequential SDP, as introduced in Sect. 2.2. In the following two subsections, we will show that friction cone constraint (8c) and contact region constraint (8d) can also be formulated as BMIs.

3.1.1 Friction Cones

We consider the Coulomb friction cones as depicted in Fig. 2. For the i^{th} friction cone $\mathcal{FC}_i$, the axis of the cone is denoted by a vector $c_i \in \mathbb{R}^3$, which is a normal vector originating at the grasp point x_i and pointing outward and perpendicular to the contact surface. We introduce c_i as a decision variable in our problem, to parameterize the friction cone $\mathcal{FC}_i$. For the benefit of the later discussion, we will constrain c_i to have unit length:

$$c_i^T c_i = 1. \tag{10}$$

If we use the nonlinear friction cone, then $f_i \in \text{int}(\mathcal{FC}_i)$ is equivalent to

$$f_i^T c_i > \frac{1}{\sqrt{\mu^2+1}}|f_i|, \tag{11a}$$

where μ is the fixed friction coefficient.

If c_i was fixed, constraint (11a) would be a *Second-order cone constraint* on f_i, which is a special type of psd constraint [1]. Thus by searching both c_i and f_i, constraints (10) and (11a) are both BMIs on variables c_i and f_i.

If $\mathcal{FC}_i$ is a linearized friction cone with n_e edges, to compute its edges, we can first construct a cone $\mathcal{FC}_0$ that has unit axis $c_0 = [0\ 0\ 1]^T$, with edges $e_0^1, e_0^2, \dots, e_0^{n_e}$. Without loss of generality we suppose all the edges of the cone have unit length, $|e_0^j| = 1$, $j = 1, \dots, n_e$. The edge e_0^j can be computed using the friction coefficient and c_0, thus they are fixed. The linearized friction cone at x_i with cone axis c_i, can be obtained by appropriately rotating cone $\mathcal{FC}_0$ such that cone axis c_0 is aligned with c_i. Such rotation is parameterized with a unit quaternion z_i, satisfying constraints:

$$z_i \otimes z_i^* = 1 \tag{12a}$$

$$c_i = R(z_i)c_0 \tag{12b}$$

where $\otimes$ is the Hamiltonian product between quaternions. z_i^* is the conjugate of z_i, and $R(z_i) \in \mathbb{R}^{3\times 3}$ is the rotation matrix corresponding to z_i, each entry in $R(z_i)$ is a second-order polynomial of z_i [21]. Applying the same rotation to the friction cone edges e_0^j generates the friction cone edges e_i^j at x_i.

$$e_i^j = R(z_i)e_0^j, \quad j = 1, \dots, n_e. \tag{13}$$

The contact force f_i is a positive weighted sum of the edges of the friction cone:

$$f_i = \sum_{j=1}^{n_e} w_i^j e_i^j, \quad w_i^j > 0 \tag{14}$$

Constraints (12a)–(14) involve only second order terms of the decision variables z_i, w_i^j, e_i^j, and thus can be posed as a BMI.

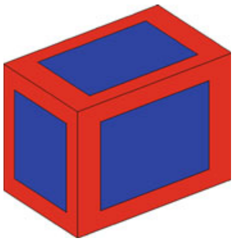


Fig. 3 The polyhedron $\mathcal{P}$ to be grasped. The admissible contact regions are the shrunk regions on each facets (*blue region*)

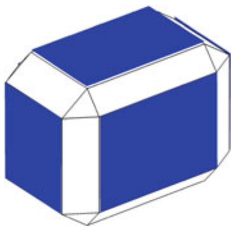


Fig. 4 The shrunk polyhedron $\mathcal{P}_s$ obtained as the convex hull of the *blue regions*, which are the shrunk regions on each facets as in Fig. 3

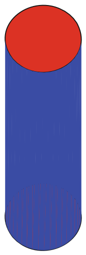


Fig. 5 The *cylinder* to be grasped, the *blue* surface is the grasp region

3.1.2 Contact Geometries

In this section, we consider four types of objects to be grasped, including convex polyhedra (Figs. 3 and 4), spheres, ellipsoids and cylinders (Fig. 5). The constraints on contact point x_i and contact normal c_i are straight-forward for the sphere, ellipsoid and cylinder, since the contact surfaces for these geometries are all parameterized by quadratic functions. Thus the constraints on x_i and c_i are also quadratic, and can be solved as BMIs. When the object is a polyhedron, and the grasp is free to choose any facets, the problem becomes trickier to handle, and we will discuss it below.

For a convex polyhedron $\mathcal{P} = \text{ConvexHull}(v_p^1, \dots, v_p^{N_p})$ (The red box in Fig. 3), where v_p^i is the i^{th} vertex of the polyhedron, we want to avoid contacts lying at edges or corners of the polyhedron, since such a grasp can be unstable and the object can slide out of the grasp. Thus the admissible contact regions are given as the shrunk surface regions (blue shades). We then construct a shrunk polyhedron (Fig. 4) as the convex hull of the shrunk surface regions (blue shades). The shrunk polyhedron is given as $\mathcal{P}_s = \{x | A_s x \le b_s\}$; this *H-representation* of a polyhedron can be readily computed from its vertices [24]. To constrain x_i lying on one of the shrunk surface regions, we use the fact that a point is on the surface of a convex object, if and only if a supporting hyperplane intersects the object at that point. Thus we introduce a supporting hyperplane $\mathcal{H}_i = \{x | c_i^T x + d_i = 0\}$, where c_i is the axis of the friction cone, and the constraints:

- The grasp point x_i is on the hyperplane

$$c_i^T x_i + d_i = 0 \tag{15}$$

- All vertices of the original polyhedron $\mathcal{P}$ lie on one side of the hyperplane, and the normal vector c_i points outward from the polyhedron

$$c_i^T v_p^j + d_i \le 0 \quad \forall j = 1, \dots, N_p \tag{16}$$

- The grasp point lies within the shrunk polyhedron $\mathcal{P}_s$

$$A_s x_i \le b_s \tag{17}$$

Geometrically, constraints (15)–(17) state that $c_i^T x + d = 0$ is a supporting hyperplane of the polyhedron $\mathcal{P}$, and the supporting point x_i is not at edges or corners of the polyhedron, so c_i has to coincide with one of the face normals. We want to highlight that we do not specify on which facet the contact lies; by searching over c_i, x_i and d_i, the optimization program will determine the contact facets by itself. Constraints (16) and (17) are linear on x_i, c_i and d_i. Constraint (15) is a BMI on x_i and c_i.

3.2 *Kinematics*

The contact points are meaningful only if they are reachable by the hand, subject to the kinematic constraints. As we will show in this section, such kinematic constraints can also be formulated as BMIs, using robot maximal coordinates.

We illustrate the kinematic chain between two links, welded by a revolute joint as in Fig. 6. The orientation of the link frame $i-1, i$ are represented by unit quaternions q_{i-1}, q_i, and the position of frame origins are $p_{i-1}, p_i \in \mathbb{R}^3$ respectively. The transformation from the axis frame i to the link frame $i-1$ is fixed, with a given unit quaternion $z_{i-1,i}$ for the rotation, and a given vector $p_{i-1,i}$ for the translation, both expressed in link frame $i-1$. We introduce two additional variables $\hat{c}_i, \hat{s}_i$, to

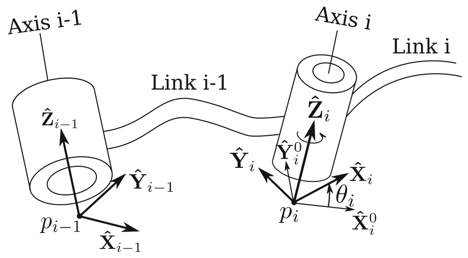


Fig. 6 [6] A link frame $\hat{\mathbf{X}}_{i-1}, \hat{\mathbf{Y}}_{i-1}, \hat{\mathbf{Z}}_{i-1}$ is attached to link $i-1$, link frame $\hat{\mathbf{X}}_i, \hat{\mathbf{Y}}_i, \hat{\mathbf{Z}}_i$ is attached to link i. $\hat{\mathbf{Z}}_{i-1}, \hat{\mathbf{Z}}_i$ are the revolute axes of the joints. The axis frame $\hat{\mathbf{X}}_i^0, \hat{\mathbf{Y}}_i^0, \hat{\mathbf{Z}}_0$ is attached to the joint i, and is fixed in the link frame $i-1$. The axis frame i and the link frame i share the frame origin and $\hat{\mathbf{Z}}_i$ axis, the latter frame is obtained by rotating the former by angle θ_i around the $\hat{\mathbf{Z}}_i$ axis

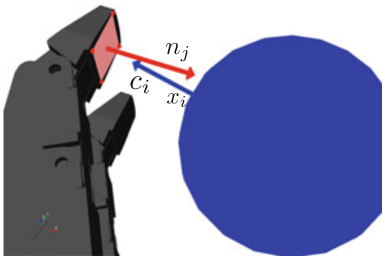


Fig. 7 To grasp the object with a given finger face (the *red shaded region* with the *red dots* as vertices) on the link j, the face normal vector n_j must be in the opposite direction of the object surface normal vector c_i, and the face must be in contact with the object at x_i

represent $\cos\frac{\theta_i}{2}$, $\sin\frac{\theta_i}{2}$ respectively. The rotation of the axis i is thus expressed by the unit quaternion $z_{\theta_i} = \hat{c}_i + \hat{s}_i\mathbf{k}$. The relationship between link frame $i-1$ and i are

$$p_i = R(q_{i-1})p_{i-1,i} + p_{i-1} \tag{18a}$$
$$q_i = q_{i-1} \otimes z_{i-1,i} \otimes z_{\theta_i} \tag{18b}$$
$$q_i \otimes q_i^* = 1,\, q_{i-1} \otimes q_{i-1}^* = 1 \tag{18c}$$

where $R(q_{i-1})$ is the rotation matrix for unit quaternion q_{i-1}.

The constraints on $\hat{c}_i, \hat{s}_i$ are

$$\hat{c}_i^2 + \hat{s}_i^2 = 1 \tag{19a}$$
$$\hat{c}_i \in \text{range}\left(\cos\frac{\theta_i}{2}\right),\ \hat{s}_i \in \text{range}\left(\sin\frac{\theta_i}{2}\right),\ \theta_i \in [\underline{\theta}_i,\ \bar{\theta}_i] \tag{19b}$$

where constraint (19a) guarantees that $\hat{c}_i, \hat{s}_i$ are the values of cosine and sine functions of a certain angle, and constraint (19b) encodes the joint limits $[\underline{\theta}_i,\ \bar{\theta}_i]$ for axis i. Constraints (18a)–(19b) encode kinematics chain that welds the adjacent links.

We constrain the hand grasping the object with a given face on link j, as shown in Fig. 7. Suppose the vertices of the contact face on link j's are given as $v_1^l, \ldots, v_{n_l}^l$ (the superscript l denotes the point measured in **link** frame), and the finger face touches the object at point x_i on the object (introduced in Sect. 3.1). By introducing extra variables α_k as convex weights, and v_k^w as the position of the k^{th} vertex in the world frame (the superscript w for **world** frame), we can express x_i as a convex combination of the finger face vertices in the world frame:

$$v_k^w = p_j + R(q_j)v_k^l \tag{20a}$$
$$\sum_{k=1}^{n_l} \alpha_k v_k^w = x_i,\ \sum_{k=1}^{n_l} \alpha_k = 1, \alpha_k \geq 0. \tag{20b}$$

Suppose the unit length face normal vector in link j's link frame is given as n_j^l. Then the face normal vector in the world frame must be the opposite to the object surface normal vector c_i, as below:

$$R(q_j)n_j^l + c_i = 0. \tag{21}$$

With kinematic constraints formulated as linear and bilinear equations in this section, we can solve the inverse kinematics problem by solving BMIs. Furthermore, we can combine the kinematic constraints and the force closure constraints in Sect. 3.1, to find a force closure grasping posture through sequential SDP.

3.3 Grasp Quality Optimization

The Q_1 metric proposed by Kirkpatrick [12, 22] measures the smallest magnitude of wrench disturbance that cannot be resisted, given an upper bound on the total contact forces. For contact point $x_i, i = 1, \dots, n$, and linearized friction cone, whose unit length edges are $e_i^j, j = 1, \dots, n_e$, we define the wrench set $\mathcal{W}$ as the set of wrench that can be resisted by those contact points, when the total contact forces on all contact points are bounded by 1.

$$\mathcal{W} = \text{Convex Hull}(V_i^j),\ i = 1, \dots, n,\ j = 1, \dots, n_e,\ \text{where } V_i^j = \begin{bmatrix} e_i^j \\ x_i \times e_i^j \end{bmatrix}. \tag{22}$$

If $\mathcal{W}$ contains the origin in the wrench space, then force closure is achieved.

The Q_1 metric is defined as the radius of the largest L_2 ball centered at the origin and being contained in the wrench set $\mathcal{W}$. An L_2 ball is formulated as $\mathcal{B} = \{w \in \mathbb{R}^6 \big| w^T Q_w w \le r^2\}$, where $Q_w \in \mathbb{R}^{6\times 6} \succeq 0$ is a given matrix (usually diagonal), which weights the relative importance between the force disturbance and the torque disturbance. We illustrate the geometric interpretation of Q_1 metric in Fig. 8. This cartoon depicts the ball and convex hull in 2D. In the real problem the wrench space has 6 dimensions.

To find the contact points and friction cones, such that their wrench set contains the largest L_2 ball, we employ an iterative procedure. In each iteration for a given L_2 ball radius r, we search contact point x_i and edges of friction cone e_i^j such that $\mathcal{B} \subset \mathcal{W}$ for that r; and then increment r in the next iteration.

To derive the condition on x_i and e_i^j such that $\mathcal{B} \subset \mathcal{W}$ for a given r, we define two cones by appending an extra dimension to $\mathcal{W}$ and $\mathcal{B}$

$$\mathcal{K}_{\mathcal{B}} = \left\{ \begin{bmatrix} w \\ t \end{bmatrix} \middle| w^T Q_w w \leq r^2 t^2, t \geq 0 \right\}, \tag{23a}$$

$$\mathcal{K}_{\mathcal{W}} = \sum_{i,j} \lambda_i^j \begin{bmatrix} V_i^j \\ 1 \end{bmatrix}, \lambda_i^j \geq 0, i = 1, \ldots, n, j = 1, \ldots, n_e. \tag{23b}$$

The cross section of cones $\mathcal{K}_{\mathcal{B}}$ and $\mathcal{K}_{\mathcal{W}}$ with plane $\left\{ \begin{bmatrix} w \\ t \end{bmatrix} \middle| t = 1 \right\}$ are $\mathcal{B}$ and $\mathcal{W}$ respectively. So the subset relation between $\mathcal{B}$, $\mathcal{W}$ is equivalent to the subset relation between $\mathcal{K}_{\mathcal{B}}$, $\mathcal{K}_{\mathcal{W}}$

$$\mathcal{B} \subset \mathcal{W} \Leftrightarrow \mathcal{K}_{\mathcal{B}} \subset \mathcal{K}_{\mathcal{W}} \Leftrightarrow \mathcal{K}_{\mathcal{W}}^* \subset \mathcal{K}_{\mathcal{B}}^*. \tag{24}$$

The second equivalence is based on the fact that for any arbitrary cones $\mathcal{K}_1, \mathcal{K}_2, \mathcal{K}_1 \subset \mathcal{K}_2 \Leftrightarrow \mathcal{K}_2^* \subset \mathcal{K}_1^*$, where $\mathcal{K}_1^*$ and $\mathcal{K}_2^*$ are the dual cones of $\mathcal{K}_1, \mathcal{K}_2$ respectively [2].

The formulation of the dual cones $\mathcal{K}_{\mathcal{W}}^*$ and $\mathcal{K}_{\mathcal{B}}^*$ are

$$\mathcal{K}_{\mathcal{W}}^* = \left\{ \begin{bmatrix} a \\ b \end{bmatrix} \middle| (V_i^j)^T a + b \geq 0, \forall i = 1, \ldots, n, j = 1, \ldots, n_e \right\} \tag{25a}$$

$$\mathcal{K}_{\mathcal{B}}^* = \left\{ \begin{bmatrix} a \\ b \end{bmatrix} \middle| b^2 \geq r^2 a^T Q_w^{-1} a, \ b \geq 0 \right\}. \tag{25b}$$

The geometric interpretation for $K_{\mathcal{W}}^* \subset K_{\mathcal{B}}^*$ is that for any half-space $\{w | w^T a + b \geq 0\}$ which contains all the vertices $V_i^j, i = 1, \ldots, n, j = 1, \ldots, n_e$, that half-space will also contain the L_2 ball $\mathcal{B}$, as shown in the cartoon Fig. 8. So the necessary and sufficient condition on $\mathcal{B} \subset \mathcal{W}$ for a given r is that

$$(V_i^j)^T a + b \geq 0, \forall i, j \ \Rightarrow \begin{cases} b^2 \geq r^2 a^T Q_w^{-1} a \\ b \geq 0 \end{cases} \tag{26}$$

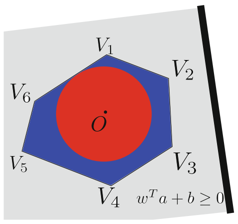


Fig. 8 The largest *ball* centered at the origin O, being contained in the convex hull spanned by vertices. A necessary and sufficient condition for the ball being contained in the convex hull, is that for any half-space parameterized as $w^T a + b \geq 0$ (the *shaded region*) that contains all the vertices of the convex hull, such half-space will also contain the ball

where $\Rightarrow$ means that the conditions on a, b on the left hand-side implies the relations on a, b on the right hand-side. We simplify this condition further to remove the quadratic term on b. Condition (26) is equivalent to

$$\begin{cases} (V_i^j)^T a + b \geq 0, \forall i, j \\ a^T Q_w^{-1} a = 1 \end{cases} \Rightarrow b \geq r. \tag{27}$$

Condition (27) is a necessary and sufficient condition for $\mathcal{B} \subset \mathcal{W}$. Using the *S-procedure* [19], we write the sufficient condition for (27) as the following algebraic constraints on polynomials, with a, b being indeterminates

$$b - r - L_1(a,b)(a^T Q_w^{-1} a - 1) - \sum_{1\leq i\leq n} \sum_{1\leq j\leq n_e} L_2^{i,j}(a,b)((V_i^j)^T a + b) \text{ is SOS} \tag{28a}$$

$$L_2^{i,j}(a,b) \text{ is SOS } \forall i, j \tag{28b}$$

where $L_1(a,b)$, $L_2^{i,j}(a,b)$ are polynomials on indeterminates a, b. SOS stands for *Sum of Squares*, and is a sufficient condition for a polynomial being non-negative. For a polynomial $\alpha(y)$ on indeterminates $y \in \mathbb{R}^k$ with highest order $2d$

$$\alpha(y) \text{ is SOS} \Leftrightarrow \alpha(y) = \Phi(y)^T H \Phi(y), H \succeq 0 \tag{29}$$

where $\Phi(y)$ is the vector containing all monomials of y up to degree d. Thus to search for a non-negative polynomial, it is sufficient to search for the psd matrix H, which ends up with a semidefinite problem on the coefficients of the polynomial. The reader can refer to [19, 23] for more details on *SOS*.

Since constraint (28a), (28b) are sufficient conditions of (27), for x_i and e_i^j satisfying constraints (28a) and (28b), r is a lower bound of its Q_1 metric. To maximize r, we can use either bilinear alternation (Algorithm 2) or binary search (Algorithm 3).

In the bilinear alternation, the k^{th} iteration is guaranteed to obtain an objective r that is at least as good as the previous iteration, since a solution to step 2 in iteration k is also feasible for step 1 in both iteration k and $k+1$; also r cannot increase to infinity. Hence the bilinear alternation will terminate with convergence of the cost. It is a common strategy in the SDP literature [14, 19, 23].

The binary search algorithm needs to deal with psd constraints of larger size than that in bilinear alternation, since it involves the product of $L_2^{i,j}(a,b)$ and V_i^j. Thus the binary search algorithm takes longer time to solve each SDP. Experimentally, we find that the binary search algorithm is less susceptible to local minima than the bilinear alternation alone.

Algorithm 2 Bilinear alternation

```
Start with a force closure grasp $x_i, e_i^j, c_i$ and $V_i^j$ found using approach described in Sects. 3.1
while $r \neg$ converged do
  1. At iteration $k$, fix $V_i^j$ in constraint (28a), search for $L_1(a,b)$, $L_2^{i,j}(a,b)$ and $r$ to maximize $r$, subject to constraints (28a), (28b). This optimization is a semi-definite programming problem. It finds an $L_2$ ball contained in the convex hull of $V_i^j$.
  2. Fix $L_2^{i,j}(a,b)$ to the solution in step 1, find feasible $V_i^j, x_i, e_i^j, c_i, L_1(a,b)$ that satisfy (28a) and constraints on $x_i, c_i, e_i^j$ in Sects. 3.1, 3.2. This is a BMI problem. It finds grasp points $x_i$ and friction cone edges $e_i^j$, such that the grasp quality is no worse than that in the previous iteration. The solution $V_i^j$ will be used in step 1 in the next iteration.
end while
```

Algorithm 3 Binary search

```
Start with $\underline{r} = 0$, and $\bar{r}$ to be some big value, $r = \frac{\bar{r}+\underline{r}}{2}$.
while $r \neg$ converged do
1. Fix $r$, search for the coefficients of $L_1(a,b)$, $L_2^{i,j}(a,b)$, $V_i^j, x_i, e_i^j, c_i$ together, subject to constraints (28a), (28b) and the constraints on $x_i, e_i^j, c_i$ in Sect. 3.1. This is a BMI problem. If the problem converges, set the lower-bound $\underline{r} = r$; otherwise set the upper-bound $\bar{r} = r$.
2. $r = \frac{\bar{r}+\underline{r}}{2}$, go to step 1.
end while
```

4 Results

4.1 Force Closure Contact

We show the results of finding force closure contact locations on different geometries in Fig. 9. We also show the time scalability w.r.t number of contacts in Figs. 10, 11, and number of polyhedron facets in Fig. 12. When we increase the number of contacts (Figs. 10 and 11), the size of the largest psd constraints remains the same, and the number of psd constraints increases linearly. As expected, the computation time in each SDP scales linearly (Fig. 11); and empirically we observe that the number of SDP calls remains almost constant (Fig. 10). As a result, the total computation time scales linearly w.r.t number of contacts. On the other hand, the number of polyhedron facets does not affect the size or the number of the psd constraints, so the total computation time remains almost constant (Fig. 12).

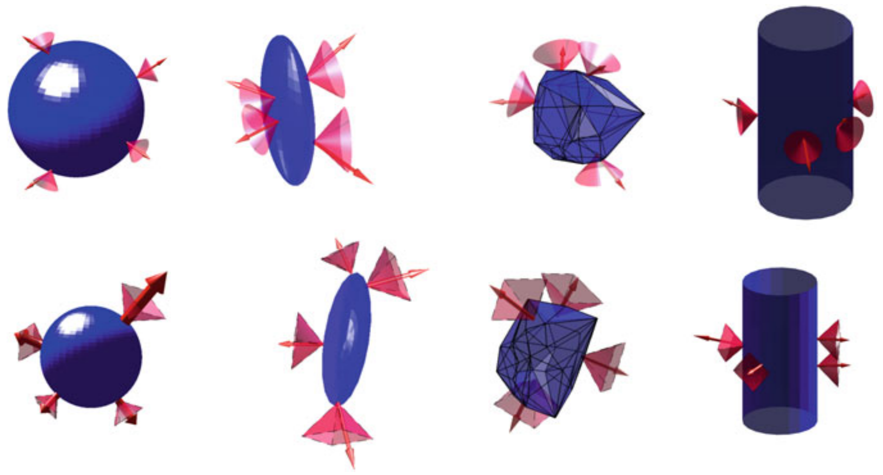
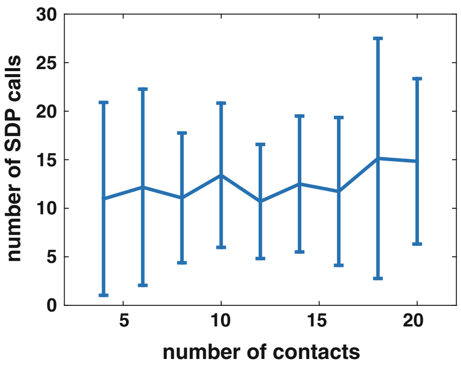


Fig. 9 Force closure contacts on different geometries. The *upper row* uses nonlinear friction cone, the *lower row* uses linearized friction cone. For the polyhedron (column 3), the contact facets are not specified by the user beforehand

Fig. 10 Scalability w.r.t number of contacts on a 30 facets polyhedron



4.2 *Kinematics*

The inverse kinematics problem is solved for an ABB IRB140 arm with a Robotiq hand, with 15 joints in total. To evaluate how effective the algorithm is as solving this inverse kinematics problem, in Fig. 13 we take 10,000 samples within the 0.6 m $\times$ 0.6 m $\times$ 0.6 m box in the shaded region, and require the center of the palm to reach the sample point. There are three possible outcomes from the sequential SDP.

- Green dot: sequential SDP converges to $t_{opt} = 0 \Rightarrow$ feasible BMIs, thus reachable.
- Red dot: the rank-relaxed SDP reports infeasibility, thus proved unreachable.
- Blue dot: the rank-relaxed SDP is feasible, but the sequential SDP does not converge to $t_{opt} = 0$.

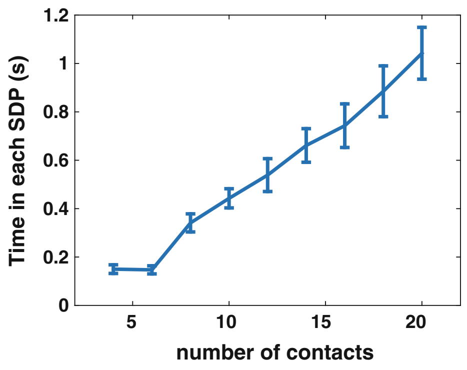


Fig. 11 Scalability w.r.t number of contacts on a 30 facets polyhedron

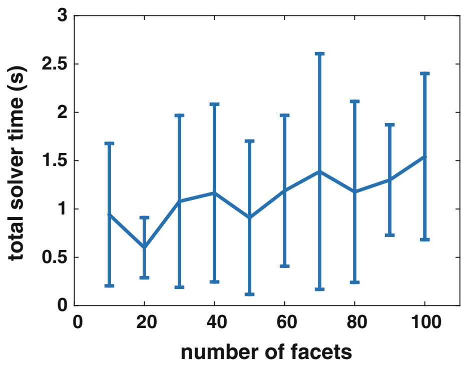


Fig. 12 Scalability w.r.t number of facets, test with 4 contacts

As shown in Fig. 13, the blue dot layer is thin, showing that in most cases the sequential SDP algorithm either solves the problem or proves that the problem is infeasible. The histogram in Fig. 14 shows that when the sequential SDP can solve the problem, in most (81.36%) cases it converges within 5 SDP calls. The average time to solve the BMI is 0.25 s using MOSEK [16] on an Intel i7 machine.

4.3 Grasp Optimization

We first show the result of using bilinear alternation to optimize a 3-point force closure grasp on a sphere. The initial contacts and linearized friction cones are plotted in Fig. 15, the optimized contacts become more evenly distributed (Fig. 16), as is known to be the better 3-point grasp on the sphere [13]. In Fig. 17 we draw the Q_1 metric in each iteration. The SOS program (28a), (28b) finds a lower bound of the Q_1 metric. The true Q_1 metric is computed as in Appendix. We can see that the gap between the SOS verified lower bound and true Q_1 metric is small. The computation time is 172 s using MOSEK solver [16] on an Intel i7 machine.

Fig. 13 Robot arm reachability

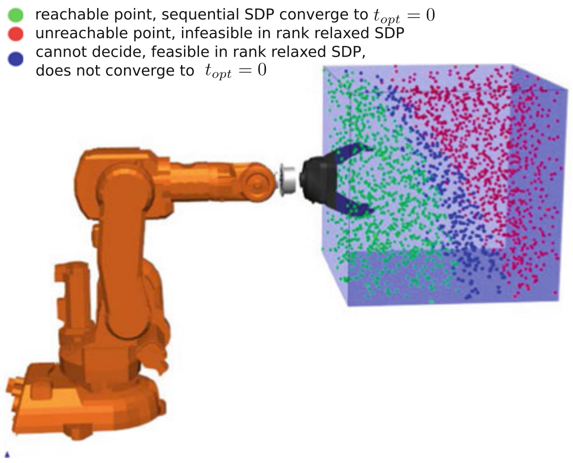

reachable point, sequential SDP converge to $t_{opt} = 0$
unreachable point, infeasible in rank relaxed SDP
cannot decide, feasible in rank relaxed SDP, does not converge to $t_{opt} = 0$

Fig. 14 Histogram on number of SDP calls

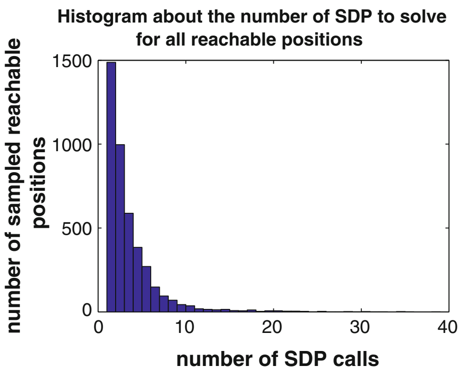


Fig. 15 Initial contacts and friction cones

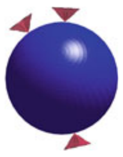
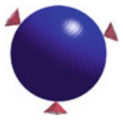


Fig. 16 Optimized contacts and friction cones



We also show the result of optimizing force closure contacts on a diamond shaped polyhedron, through binary search. The optimized contacts (Fig. 19) are more evenly distributed than the initial contacts (Fig. 18). Also we want to highlight that the facets on which the contacts lie are changed through optimization, this again demonstrates that the optimization program can search over *all* facets by itself. The computation time is around an hour using MOSEK solver on an Intel i7 machine (Fig. 20).

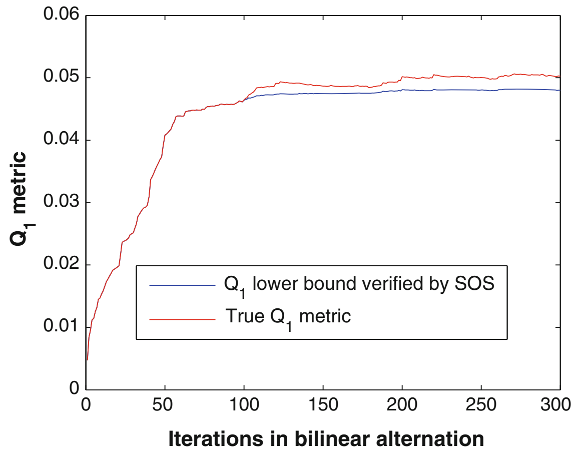


Fig. 17 The change of Q_1 metric in each iteration of bilinear alternation

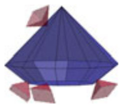


Fig. 18 Initial contacts and friction cones

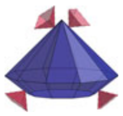


Fig. 19 Optimized contacts and friction cones

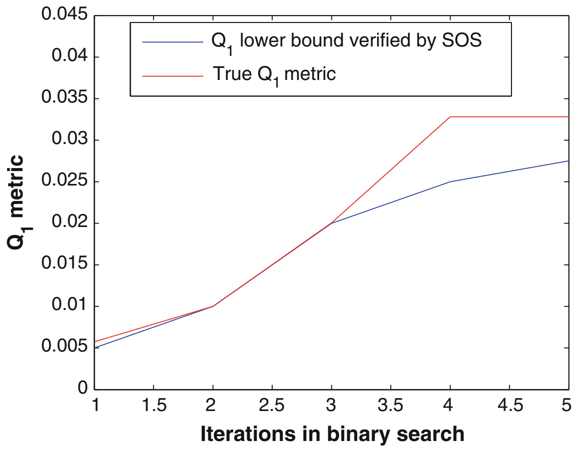


Fig. 20 The change of Q_1 metric in each iteration of binary search

We show the result of optimizing the force closure grasp with Robotiq hand and ABB arm on a cylinder. The initial posture grasps the tip of the cylinder (Fig. 21), the optimized posture gets improved by grasping the center of the cylinder (Fig. 22). The computation time is around 20 min using MOSEK on an Intel i7 machine (Fig. 23).

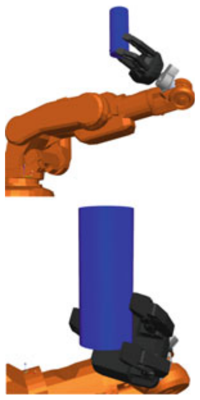


Fig. 21 Initial force closure grasp from two views

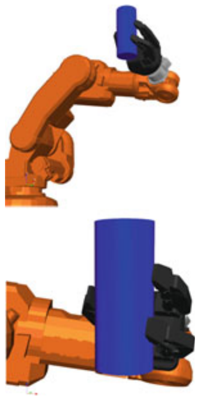


Fig. 22 Optimized force closure grasp from two views

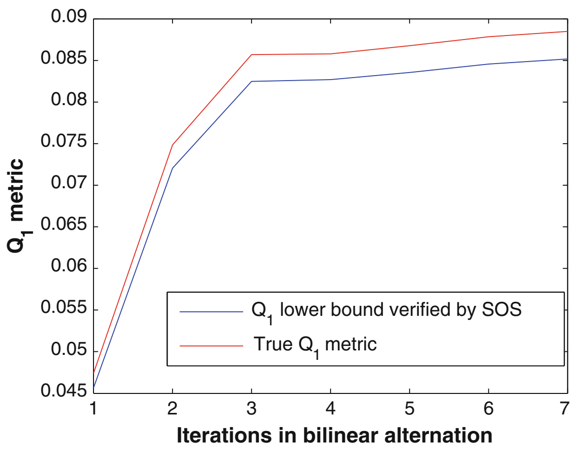


Fig. 23 The change of Q_1 metric in each iteration of bilinear alternation, for robotiq hand grasping the cylinder

5 Conclusion and Discussion

In this paper we exploit the bilinear structure in the force closure and kinematic constraints to synthesize and optimize force closure grasping postures. We do this by formulating the problem as bilinear matrix inequalities (BMIs) and applying the sequential semidefinite programming technique commonly employed in the BMI literature. In contrast to more conventional approaches to the problem that rely on gradient based nonlinear optimization, our approach is able to handle non-smooth (such as psd) constraints along with being able to prove *infeasibility* of problems. We demonstrate our results on a 15-joint robot and several types of object geometries.

Some tangible improvements include using the nonlinear friction cone when optimizing force closure grasps, dealing with non-convex polyhedron object, etc.

Acknowledgements We would like to thank Amir Ali Ahmadi for introducing BMI and reference [11]; Alberto Rodriguez for the discussion, and ABB Inc for loaning the IRB140 arm.

Appendix

When all contact points x_i and friction cone edges e_i^j are given, we can compute the exact value of Q_1 metric. First we transform representation of the wrench set $\mathcal{W}$ from using vertices V_i^j (*V-representation*) to using half-spaces (*H-representation*) $\mathcal{W} = \left\{w | w^T a_{\mathcal{W}}^i \le b_{\mathcal{W}}^i,\ i = 1, \ldots, m\right\}$, where m is the number of facets for $\mathcal{W}$. The Q_1 metric is computed as $\min_{i=1,\ldots,m} b_{\mathcal{W}}^i / \sqrt{\left(a_{\mathcal{W}}^i\right)^T Q_w^{-1} a_{\mathcal{W}}}$. Note that we cannot optimize the Q_1 metric while searching for x_i and e_i^j, since it is nontrivial to transform from *V-representation* to *H-representation* when the vertices are not fixed [24].

References

1. Alizadeh, F., Goldfarb, D.: Second-order cone programming. Math. Program. **95**(1), 3–51 (2003)
2. Boyd, S., Vandenberghe, L.: Convex Optimization. Cambridge University Press, Cambridge (2004)
3. Boyd, S.P., Wegbreit, B.: Fast computation of optimal contact forces. IEEE Trans. Robot. **23**(6), 1117–1132 (2007)
4. Buss, S.R.: Introduction to inverse kinematics with jacobian transpose, pseudoinverse and damped least squares methods (2004)
5. Chen, I.-M., Burdick, J.W.: Finding antipodal point grasps on irregularly shaped objects. IEEE Trans. Robot. Autom. **9**(4), 507–512 (1993)
6. Craig, J.J.: Introduction to Robotics: Mechanics and Control, 3rd edn. Pearson Education Inc, New Jersey (2005)
7. Dai, H., Valenzuela, A., Tedrake, R.: Whole-body motion planning with centroidal dynamics and full kinematics. In: IEEE-RAS International Conference on Humanoid Robots (2014)

8. Fazel, M.: Matrix rank minimization with applications. Ph.D. thesis (2002)
9. Ferrari, C., Canny, J.: Planning optimal grasps. In: 1992 IEEE International Conference on Robotics and Automation, 1992. Proceedings, pp. 2290–2295. IEEE (1992)
10. Han, L., Trinkle, J.C., Li, Z.X.: Grasp analysis as linear matrix inequality problems. IEEE Trans. Robot. Autom. **16**(6), 663–674 (2000)
11. Ibaraki, S., Tomizuka, M.: Rank minimization approach for solving bmi problems with random search. In: Proceedings of the 2001. American Control Conference, 2001, vol. 3, pp. 1870–1875. IEEE (2001)
12. Kirkpatrick, D., Mishra, B., Yap, C.-K.: Quantitative steinitz's theorems with applications to multifingered grasping. Discret. Comput. Geom. **7**(1), 295–318 (1992)
13. Liu, G., Xu, J., Wang, X., Li, Z.: On quality functions for grasp synthesis, fixture planning, and coordinated manipulation. IEEE Trans. Autom. Sci. Eng. **1**(2), 146–162 (2004)
14. Majumdar, A., Ahmadi, A.A., Tedrake, R.: Control design along trajectories with sums of squares programming. In: Proceedings of the 2013 IEEE International Conference on Robotics and Automation (ICRA) (2013)
15. Mishra, B.: Grasp metrics: optimality and complexity. In: Proceedings of the workshop on Algorithmic foundations of robotics, pp. 137–165. AK Peters, Ltd. (1995)
16. Mosek, A.: The mosek optimization software. Online at: http://www.mosek.com, vol. 54 (2010)
17. Murray, R.M., Li, Z., Sastry, S.S.: A Mathematical Introduction to Robotic Manipulation. CRC Press Inc., Boca Raton (1994)
18. Nguyen, V.-D.: Constructing force-closure grasps. Int. J. Robot. Res. **7**(3), 3–16 (1988)
19. Parrilo, P.A.: Structured Semidefinite Programs and Semialgebraic Geometry Methods in Robustness and Optimization. Ph.D. thesis, California Institute of Technology, May 18 2000
20. Rosales, C., Ros, L., Porta, J.M., Suárez, R.: Synthesizing grasp configurations with specified contact regions. Int. J. Robot. Res. **30**(4), 431–443 (2011)
21. Salamin, E.: Application of quaternions to computation with rotations. Technical report, Working Paper (1979)
22. Schulman, J.D., Goldberg, K., Abbeel, P.: Grasping and fixturing as submodular coverage problems. In: International Symposium on Robotics Research, pp. 1–12 (2011)
23. Tedrake, R., Manchester, I.R., Tobenkin, M.M., Roberts, J.W.: LQR-Trees: feedback motion planning via sums of squares verification. Int. J. Robot. Res. **29**, 1038–1052 (2010)
24. Ziegler, G.: Lectures on polytopes, vol. 152. Springer Science and Business Media (1995)

Using Geometry to Detect Grasp Poses in 3D Point Clouds

Andreas ten Pas and Robert Platt

1 Introduction

Traditionally, robot grasping is understood in terms of two related subproblems: perception and planning. The goal of the perceptual component is to estimate the position and orientation (pose) of an object to be grasped. Then, grasp and motion planners are used to calculate where to move the robot arm and hand in order to perform grasp. While this approach can work in ideal scenarios, it has proven to be surprisingly difficult to localize the pose of novel objects in clutter accurately [6]. More recently, researchers have proposed various grasp point detection methods that localize grasps independently of object identity. One class of approaches use a sliding window to detect regions of an RGBD image or a height map where a grasp is likely to succeed [4, 5, 8, 10, 12, 16]. Other approaches extrapolate local "grasp prototypes" based on human-provided grasp demonstrations [3, 7, 11].

A missing element in the above works is that they do not leverage the geometry of grasping to improve detection. Grasp geometry has been studied extensively in the literature (for example [13, 17]). Moreover, point clouds created using depth sensors would seem to be well suited for geometric reasoning. In this paper, we propose an algorithm that detects grasps in a point cloud by predicting the presence of necessary and sufficient geometric conditions for grasping. The algorithm has two steps. First, we sample a large set of grasp hypotheses. Then, we classify those hypotheses as grasps or not using machine learning. Geometric information is used in both steps. First, we use geometry to reduce the size of the sample space. A trivial necessary condition for a grasp to exist is that the hand must be collision-free and part of the object surface must be contained between the two fingers. We propose

A. ten Pas (✉) · R. Platt
Northeastern University, Boston, USA
e-mail: atp@ccs.neu.edu

R. Platt
e-mail: rplatt@ccs.neu.edu

A. Bicchi and W. Burgard (eds.), *Robotics Research*, Springer Proceedings
in Advanced Robotics 2, DOI 10.1007/978-3-319-51532-8_19

a sampling method that only produces hypotheses that satisfy this condition. This simple step should boost detection accuracy relative to approaches that consider every possible hand placement a valid hypothesis. The second way that our algorithm uses geometric information is to automatically label the training set. A necessary and sufficient condition for a two-finger grasp is an antipodal contact configuration (see Definition 1). Unfortunately, we cannot reliably detect an antipodal configuration in most real-world point clouds because of occlusions. However, it is nevertheless possible *sometimes* to verify a grasp using this condition. We use the antipodal condition to label a subset of grasp hypotheses in arbitrary point clouds containing ordinary graspable objects. We generate large amounts of training data this way because it is relatively easy to take lots of range images of ordinary objects. This is a huge advantage relative to approaches that depend on human annotations because large amounts of training data can significantly improve classification performance.

Our experiments indicate that the approach described above performs well in practice. We find that without using any machine learning and just using our collision-free sampling algorithm as a grasp detection method, we achieve a 73% grasp success rate for novel objects. This is remarkable because this is a trivially simple detection criterion. When a classification step is added to the process, our grasp success rate jumps to 88%. This success rate is competitive with the best results that have been reported. However, what is particularly interesting is the fact that our algorithm achieves an average 73% grasp success rate in dense clutter such as that shown in Fig. 1. This is exciting because dense clutter is a worst-case scenario for grasping. Clutter creates lots of occlusions that make perception more difficult and obstacles that make reaching and grasping harder.

1.1 Related Work

The idea of searching an image for grasp targets independently of object identity was probably explored first in Saxena's early work that used a sliding window classifier to localize good grasps based on a broad collection of local visual features [16]. Later work extended this concept to range data [8] and explored a deep learning approach [12]. In [12], they obtain an 84% success rate on Baxter and a 92%

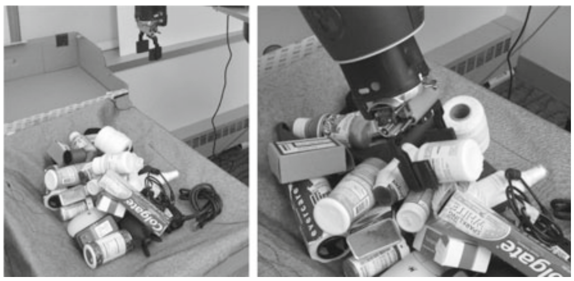


Fig. 1 Our algorithm is able to localize and grasp novel objects in dense clutter

success rate on the PR2 for objects presented in isolation (averaged over 100 trials). Fischinger and Vincze developed a similar method that uses heightmaps instead of range images and develops a different Haar-like feature representation [4, 5]. In [5], they report a 92% single-object grasp success rate averaged over 50 grasp trials using the PR2. This work is particularly interesting because they demonstrate clutter results where the robot grasps and removes up to 10 piled objects from a box. They report that over six clear-the-box runs, their algorithm removes an average of 87% of the objects from the box. Other approaches search a range image or point cloud for hand-coded geometries that are expected to be associated with a good grasp. For example Klingbeil et al. search a range image for a gripper-shaped pattern [10]. In our prior work, we developed an approach to localizing handles by searching a point cloud for a cylindrical shell [19]. Other approaches follow a template-based approach where grasps that are demonstrated on a set of training objects are generalized to new objects. For example, Herzog et al. learn to select a grasp template from a library based on features of the novel object [7]. Detry et al. grasp novel objects by modeling the geometry of local object shapes and fitting these shapes to new objects [3]. Kroemer et al. propose an object affordance learning strategy where the system learns to match shape templates against various actions afforded by those templates [11]. Another class of approaches worth mentioning are based on interacting with a stack of objects. For example, Katz et al. developed a method of grasping novel objects based on interactively pushing the objects in order to improve object segmentation [9]. Chang et al. developed a method of segmenting objects by physically manipulating them [2]. The approach presented in this paper is distinguished from the above primarily because of the way we use geometric information. Our use of geometry to generate grasp hypotheses is novel. Moreover, our ability to generate large amounts of labeled training data could be very important for improving detection accuracy in the future. However, what is perhaps most important is that we demonstrate "reasonable" (73%) grasp success rates in dense clutter – arguably a worst-case scenario for grasping.

2 Approach

We frame the problem of localizing grasp targets in terms of locating *antipodal hands*, an idea that we introduce based on the concept of an antipodal grasp. In an antipodal grasp, the robot hand is able to apply opposite and co-linear forces at two points:

Definition 1 (*Nguyen* [14]) A pair of point contacts with friction is **antipodal** if and only if the line connecting the contact points lies inside both friction cones.[1]

[1]A friction cone describes the space of normal and frictional forces that a point contact with friction can apply to the contacted surface [13].

If an antipodal grasp exists, then the robot can hold the object by applying sufficiently large forces along the line connecting the two contact points. In this paper, we restrict consideration to parallel jaw grippers – hands with parallel fingers and a single closing degree of freedom. Since a parallel jaw gripper can only apply forces along the (single) direction of gripper motion, we will additionally require the two contact points to lie along a line parallel to the direction of finger motion. Rather than localizing antipodal contact configurations directly, we will localize hand configurations where we expect an antipodal grasp to be achieved in the future when the hand closes. Let $\mathscr{W} \subseteq \mathbb{R}^3$ denote the robot workspace and let $\mathscr{O} \subseteq \mathscr{W}$ denote space occupied by objects or obstacles. Let $H \subseteq SE(3)$ denote the configuration space of the hand when the fingers are fully open. We will refer to a configuration $h \in H$ as simply a "hand". Let $B(h) \subseteq W$ denote the volume occupied by the hand in configuration $h \in H$, when the fingers are fully open.

Definition 2 An **antipodal hand** is a pose of the hand, $h \in H$, such that the hand is not in collision with any objects or obstacles, $B(h) \cap \mathscr{O} = \emptyset$, and at least one pair of antipodal contacts will be formed when the fingers close such that the line connecting the two contacts is parallel to the direction of finger motion.

Algorithm 1 illustrates at a high level our algorithm for detecting antipodal hands. It takes a point cloud, $\mathscr{C} \subseteq \mathbb{R}^3$, and a geometric model of the robot hand as input and produces as output a set of hands, $\mathscr{H} \subseteq H$, that are predicted to be antipodal. There are two main steps. First, we sample a set of hand hypotheses. Then, we classify each hypothesis as an antipodal hand or not. These steps are described in detail in the following sections.

Algorithm 1 Detect_Antipodal_Hands

```
Input: a point cloud, $\mathscr{C}$, and hand parameters, $\theta$
Output: antipodal hands, $\mathscr{H}$
1: $\mathscr{H}_{hyp} = Sample\_Hands(\mathscr{C})$
2: $\mathscr{H} = Classify\_Hands(\mathscr{H}_{hyp})$
```

3 Sampling Hands

A key part of our algorithm is the approach to sampling from the space of hand hypotheses. A naive approach would be to sample directly from $H \subseteq SE(3)$. Unfortunately, this would be immensely inefficient because $SE(3)$ is a 6-DOF space and many hands sampled this way would be far away from any visible parts of the point cloud. Instead, we define a lower-dimensional sample space constrained by the geometry of the point cloud.

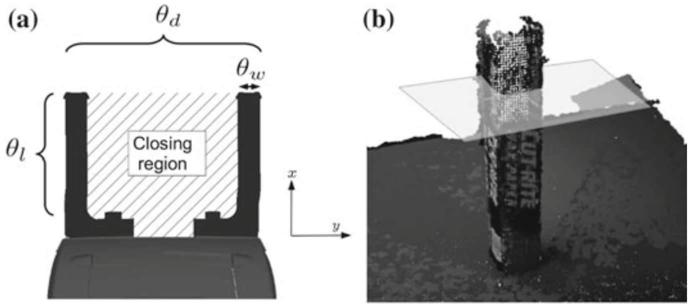


Fig. 2 **a** Hand geometry. **b** Cutting plane geometry

3.1 Geometry of the Hand and the Object Surface

Before describing the sample space, we quantify certain parameters related to the grasp geometry. We assume the hand, $h \in H$, is a parallel jaw gripper comprised of two parallel fingers each modeled as a rectangular prism that moves parallel to a common plane. Let $\hat{a}(h)$ denote a unit vector orthogonal to this plane. The hand is fully specified by the parameter vector $\theta = (\theta_l, \theta_w, \theta_d, \theta_t)$ where θ_l and θ_w denote the length and width of the fingers; θ_d denotes the distance between the fingers when fully open; and θ_t denotes the thickness of the fingers (orthogonal to the page in Fig. 2a). Define the **closing region**, $R(h) \subseteq \mathscr{W}$, to be the volumetric region swept out by the fingers when they close. Let $r(h) \in R(h)$ denote an arbitrary reference point in the closing region. Define the **closing plane**, $C(h)$, to be the subset of the plane that intersects $r(h)$, is orthogonal to $\hat{a}(h)$, and is contained within $R(h)$: $C(h) = \{p \in R(h) | (p - r(h))^T \hat{a}(h) = 0\}$.

We also introduce some notation related to the differential geometry of the surfaces we are grasping. Recall that each point on a differentiable surface is associated with a surface normal and two principal curvatures where each principal curvature is associated with a principal direction. The surface normal and the two principal directions define an orthogonal basis known as a Darboux frame.[2] The Darboux frame at point $p \in \mathscr{C}$ will be denoted: $F(p) = (\hat{n}(p)\ (\hat{a}(p) \times \hat{n}(p))\ \hat{a}(p))$, where $\hat{n}(p)$ denotes the unit surface normal and $\hat{a}(p)$ denotes the direction of minimum principal curvature at point p. Define the **cutting plane** to be the plane orthogonal to $\hat{a}(p)$ that passes through p (see Fig. 2b). Since we are dealing with point clouds, it is not possible to measure the Darboux frame exactly at each point. Instead, we estimate the surface normal and principle directions over a small neighborhood. We fit a

[2]Any frame aligned with the surface normal is a Darboux frame. Here we restrict consideration to the special case where it is also aligned with the principal directions.

quadratic function over the points contained within a small ball (3 cm radius in our experiments) using Taubin's method [18, 19] and use that to calculate the Darboux frame.[3]

3.2 Hand Sample Set

We want a set that contains many antipodal hands and from which it is easy to draw samples. The following conditions define the set $\mathcal{H}$. First, for every hand, $h \in \mathcal{H}$:

Constraint 1 *The body of the hand is not in collision with the point cloud:* $B(h) \cap \mathcal{C} = \emptyset$.

Furthermore, there must exist a point in the cloud, $p \in \mathcal{C}$, such that:

Constraint 2 *The hand closing plane contains p:* $p \in C(h)$.

Constraint 3 *The closing plane of the hand is parallel to the cutting plane at p:* $\hat{a}(p) = \hat{a}(h)$.

These three constraints define the following set of hands:

$$\mathcal{H} = \cup_{p \in \mathcal{C}} H(p), \;\; H(p) = \{h \in H | p \in C(h) \wedge \hat{a}(p) = \hat{a}(h) \wedge B(h) \cap \mathcal{C} = \emptyset\}. \tag{1}$$

Constraint 3 is essentially a heuristic that limits the hand hypotheses that our algorithm considers. While this eliminates from consideration many otherwise good grasps, it is a practical way to focus detection on likely candidates. One motivation behind this constraint is that humans prefer grasps for which the wrist is oriented orthogonally to one of the object's principal axes [1]. Moreover, it is easy to sample from $\mathcal{H}$ by: (1) sampling a point, $p \in \mathcal{C}$, from the cloud; (2) sampling one or more hands from $H(p)$. Notice that for each $p \in \mathcal{C}$, $H(p)$ is three-DOF because we have constrained two DOF of orientation and one DOF of position. This means that $\mathcal{H}$ is much smaller than H and it can therefore be covered by many fewer samples.

The sampling process is detailed in Algorithm 2. First, we preprocess the point cloud, $\mathcal{C}$, in the usual way by voxelizing (we use voxels 3 mm on a side in our experiments) and applying workspace limits (Step 2). Second, we iteratively sample a set of n points (n is between 4000 and 8000 in our experiments) from the cloud (Step 4). For each point, $p \in \mathcal{C}$, we calculate a neighborhood, $N(p)$, in the θ_d-ball around p (using a KD-tree, Step 5). The next step is to estimate the Darboux frame at p by fitting a quadratic surface using Taubin's method and calculating the surface normal and principal curvature directions (Step 6). Next, we sample a set of hand configurations

[3]Taubin's method is an analytic solution that performs this fit efficiently by solving a generalized Eigenvalue problem on two 10×10 matrices [18]. In comparison to using first order estimates of surface normal and curvature, the estimates derived from this quadratic are more robust to local surface discontinuities.

Algorithm 2 Sample_Hands

```
Input: point cloud, $\mathscr{C}$, hand parameters, $\theta$
Output: grasp hypotheses, $\mathscr{H}$
1: $\mathscr{H} = \emptyset$
2: Preprocess $\mathscr{C}$ (voxelize; workspace limits; etc.)
3: for i = 1 to n do
4:   Sample $p \in \mathscr{C}$ uniformly randomly
5:   Calculate $\theta_d$-ball about $p$: $N(p) = \{q \in \mathscr{C} : \|p - q\| \leq \theta_d\}$
6:   Estimate local Darboux frame at $p$: $F(p) = Estimate\_Darboux(N(p))$
7:   $H = Grid\_Search(F(p), N(p))$
8:   $\mathscr{H} = \mathscr{H} \cup H$
9: end for
```

over a coarse two-DOF grid in a neighborhood about p. Let $h_{x,y,\phi}(p) \in H(p)$ denote the hand at position $(x, y, 0)$ with orientation ϕ with respect to the Darboux frame, $F(p)$. Let Φ denote a discrete set of orientations (8 in our implementation). Let X denote a discrete set of hand positions (20 in our implementation). For each hand configuration $(\phi, x) \in \Phi \times X$, we calculate the hand configuration furthest along the y axis that remains collision free: $y^* = \max_{y \in Y}$ such that $B(h_{x,y,\phi}) \cap N = \emptyset$, where $Y = [-\theta_d, \theta_d]$ (Step 3). Then, we check whether the closing plane for this hand configuration contains points in the cloud (Step 4). If it does, then we add the hand to the hypotheses set (Step 5).

Algorithm 3 Grid_Search

```
Input: neighborhood point cloud, $N$; Darboux frame, $F$
Output: neighborhood grasp hypotheses, $H$
1: $H = \emptyset$
2: for all $(\phi, x) \in \Phi \times X$ do
3:   Push hand until collision: $y^* = \max_{y \in Y}$ such that $B(h_{\phi,x,y}) \cap N = \emptyset$
4:   if closing plane not empty: $C(h_{\phi,x,y^*}) \cap N \neq \emptyset$ then
5:     $H = H \cup h_{\phi,x,y^*}$
6:   end if
7: end for
```

4 Classifying Hand Hypotheses

After generating hand hypotheses, the next step is to classify each of those hypotheses as antipodal or not. The simplest approach would be to infer object surface geometry from the point cloud and then check which hands satisfy Definition 2. Unfortunately, since most real-world point clouds are partial, many hand hypotheses will fail this check simply because all relevant object surfaces were not visible to a sensor. Instead, we infer which hypotheses are likely to be antipodal using machine learning (i.e. classification).

4.1 Labeling Grasp Hypotheses

Many approaches to grasp point detection require large amounts of training data where humans have annotated images with good grasp points [4, 5, 7, 8, 12, 16]. Unfortunately, obtaining these labels is challenging because it can be hard for human labelers to predict what object surfaces in a scene might be graspable for a robot. Instead, our method automatically labels a set of training images by checking a relaxed version of the conditions of Definition 2.

In order to check whether a hand hypotheses, $h \in H$, is antipodal, we need to determine whether an antipodal pair of contacts will be formed when the hand closes. Let $\hat{f}(h)$ denote the direction of closing of one finger. (In a parallel jaw gripper, the other finger closes in the opposite direction). When the fingers close, they will make first contact with an extremal pair of points, $s_1, s_2 \in R(h)$ such that $\forall s \in R(h), s_1^T \hat{f}(h) \geq s^T \hat{f}(h) \wedge s_2^T \hat{f}(h) \leq s^T \hat{f}(h)$. An antipodal hand requires two such extremal points to be antipodal and for the line connecting the points to be parallel to the direction of finger closing. In practice, we relax this condition slightly as follows. First, rather than checking for extremal points, we check for points that have a surface normal parallel to the direction of closing. This is essentially a first-order condition for an extremal point that is more robust to outliers in the cloud. The second way that we relax Definition 2 is to drop the requirement that the line connecting the two contacts be parallel to the direction of finger closing and to substitute a requirement that at least k points are found with an appropriate surface normal. Again, the intention here is to make detection more robust: if there are at least k points near each finger with surface normals parallel to the direction of closing, then it is likely that the line connecting at least one pair will be nearly parallel to the direction of finger closing. In summary, we check whether the following definition is satisfied:

Definition 3 A hand, $h \in H$, is **near antipodal** for thresholds $k \in \mathbb{N}$ and $\theta \in [0, pi/2]$ when there exist k points $p_1, \ldots, p_k \in R(h) \cap \mathscr{C}$ such that $\hat{n}(p_i)^T \hat{f}(h) \geq \cos\theta$ and k points $q_1, \ldots, q_k \in R(h) \cap \mathscr{C}$ such that $\hat{n}(q_i)^T \hat{f}(h) \leq -\cos\theta$.

When Definition 3 is satisfied, then we label the corresponding hand a positive instance. Note that in order to check for this condition, it is necessary to register at least two point clouds produced by range sensors that have observed the scene from different perspectives (Fig. 3). This is because we need to “see” two nearly opposite surfaces on an object. Even then, many antipodal hands will not be identified as such because only one side of the object is visible. These “indeterminate” hands are omitted from the training set. In some cases, it is possible to verify that a particular hand is *not* antipodal by checking that there are fewer than k points in the hand

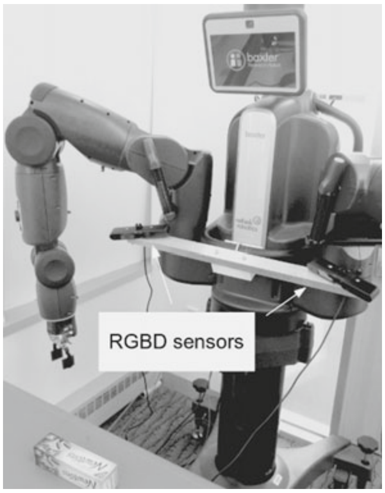


Fig. 3 Our robot has stereo RGBD sensors

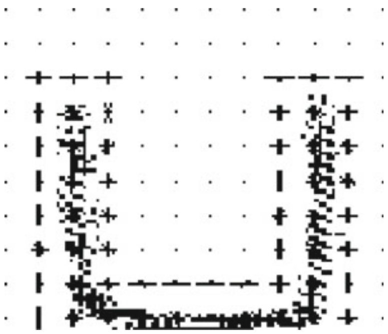


Fig. 4 HOG feature representation of a hand hypothesis for the box shown in Fig. 2b

closing region that satisfy either of the conditions of Definition 3. These hands are included in the training set as negative examples. This assumes that the closing region of every sampled hand hypothesis is at least partially visible to a sensor. If there are fewer than k satisfying points, then Definition 3 would not be satisfied even if the opposite side of an object was observed. In our experiments, we set the thresholds $k = 6$ and $\theta = 20$ degrees. However, our qualitative results are not terribly sensitive to these exact numbers. In general, it might be necessary to tune these parameters (especially k) with respect to the number of points that can be expected to be found on a graspable object and the accuracy of the surface normal estimates (Fig. 4).

4.2 Feature Representation

In order to classify hand hypotheses, a feature descriptor is needed. Specifically, for a given hand $h \in H$, we need to encode the geometry of the points contained within the hand closing region, $\mathscr{C} \cap R(h)$. A variety of relevant descriptors have been explored in the literature [15, 20]. In our case, we achieve good performance using a simple descriptor based on HOG features. For a point cloud, $\mathscr{C}$, a two dimensional image of the closing region is created by projecting the points $\mathscr{C} \cap R(h)$ onto the hand closing plane: $I(\mathscr{C}, h) = S_{12}F(h)^T \, (N \cap C(h))$, where $S_{12} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix}$ selects the first two rows of $F(h)^T$. We call this the **grasp hypothesis image**. We encode it using the HOG descriptor, $HOG(I(\mathscr{C}, h))$. In our implementation, we chose a HOG cell size such that the grasp hypothesis image was covered by 10×12 cells with a standard 2×2 block size.

4.3 Creating the Training Set

In order to create the training set, we obtain a set of objects that have local geometries similar to what might be expected in the field. In our work, we selected the set of 18 objects shown in Fig. 5a. Each object was placed in front of the robot in two configurations: one upright configuration and one on its side. For each configuration (36 configurations total), let $\mathscr{C}_1$ and $\mathscr{C}_2$ denote the voxelized point clouds obtained from each of the two sensors, respectively, and let $\mathscr{C}_{12} = \mathscr{C}_1 \cup \mathscr{C}_2$ denote the registered two-view cloud.

The training data is generated as follows. First, we extract hand hypotheses from the registered cloud, $\mathscr{C}_{12}$ using the methods of Sect. 3. Second, for each $h \in H$, we determine whether it is a positive, negative, or indeterminate by checking the conditions of Definition 3. Indeterminate hands are discarded from training. Third, for each positive or negative hand, we extract three feature descriptors: $HOG(I(\mathscr{C}_1, h))$, $HOG(I(\mathscr{C}_2, h))$, and $HOG(I(\mathscr{C}_{12}, h))$. Each descriptor is given the same label and incorporated into the training set. Over our 18 object training set, this procedure generated approximately 6500 positive and negative labeled examples that were used to train an SVM. We only did one round of training using this single training set.

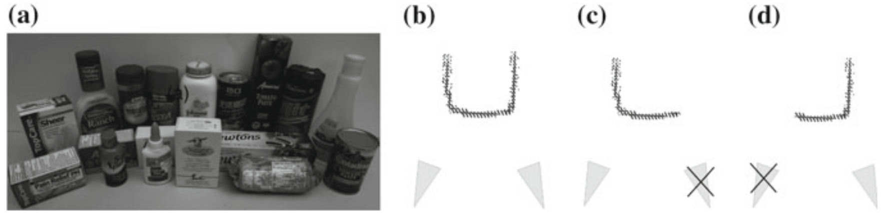


Fig. 5 **a** Training set comprised of 18 objects. **b**–**d** Illustration of the three grasp hypotheses images incorporated into the training set per hand. The *blue triangles* at the *bottom* denote positions of the two range sensors. **c**, **d** Illustrate training images created using data from only one sensor

The fact that we extract three feature descriptors per hand in step three above is important because it helps us to capture the appearance of partial views in the training set. Figure 5b–d illustrates the three descriptors for an antipodal hand. Even though the closing region of this hand is relatively well observed in $\mathscr{C}_{12}$, the fact that we incorporate $HOG(I(\mathscr{C}_1, h))$ and $HOG(I(\mathscr{C}_2, h))$ into the dataset means that we are emulating what *would* have been observed if we only had a partial view. This makes our method much more robust to partial point cloud information.

4.4 Cross Validation

We performed cross validation on a dataset derived from the 18 training objects shown in Fig. 5a. For each object, we obtained a registered point cloud for two configurations (total of 36 configurations). Following the procedure described in this section, we obtained 6500 labeled features with 3405 positives and 3095 negatives. We did 10-fold cross validation on this dataset using an SVM for the various Gaussian and polynomial kernels available in Matlab. We obtained 97.8% accuracy using a degree-three polynomial kernel and used this kernel in the remainder of our experiments.

In the cross validation experiment described above, the folds were random across the labeled pairs in the dataset. This does not capture the effects of experiencing novel objects or the expected performance when only single-view point clouds are available. Therefore, we did the following. First, we trained the system using the degree-three polynomial kernel on the 6500 labeled examples as described above. Then, we obtained additional single-view point clouds for each of the 30 novel test objects shown in Fig. 6 (each object was presented in isolation) for a total of 122 single-view points clouds. We used the methods described in this section to obtain ground-truth for this dataset. This gave us a total of 7250 labeled single-view hypotheses on novel objects with 1130 positives and 6120 negatives. We obtained 94.3% accuracy on this dataset. The fact that we do relatively well in these cross

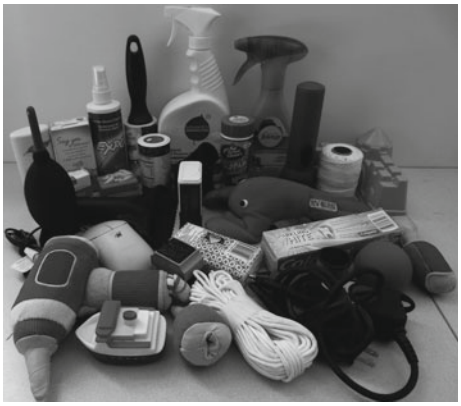


Fig. 6 The 30 objects in our test set

validation experiments using a relatively simple feature descriptor and without mining hard negatives suggests that our approach to sampling hands and creating the grasp hypothesis image makes the grasp classification task easier than it is in approaches that do not use this kind of structure [4, 5, 8, 12, 16].

5 Robot Experiments

We evaluated the performance of our algorithms using the Baxter robot from Rethink Robotics. We explore two experimental settings: when objects are presented to the robot in isolation and when objects are presented in a dense clutter scenario. We use the Baxter right arm equipped with the stock two-finger Baxter gripper. A key constraint of the Baxter gripper is the limited finger stroke: each finger has only 2 cm stroke. In these experiments, we adjust the finger positions such that they are 3 cm apart when closed and 7 cm apart when open. This means we cannot grasp anything smaller than 3 cm or larger than 7 cm. We chose each object in the training and test sets so that it could be grasped under these constraints. Two-view registered point clouds were created using Asus Xtion Pro range sensors (see Fig. 3). We implemented our algorithm in C++ on an Intel i7 3.5 GHz system (four physical CPU cores) with 16GB of system memory. On average, learning the SVM model on the 18 object training set shown in Fig. 5a takes about five minutes, while online grasp detection and selection given a single point cloud takes about 2.7 s (with 4000 hand samples). It should be possible for anyone with a Baxter robot and the appropriate depth sensors to replicate any of these experiments by running our ROS package at http://wiki.ros.org/agile_grasp.

5.1 Grasp Selection

Since our algorithm typically finds tens or hundreds of potential antipodal hands, depending upon the number of objects in the scene, it is necessary to select one to execute. One method might be to select a grasp on an object of interest. However, in this paper, we ignore object identity and perform any feasible grasp. We choose a grasp to attempt as follows. First, we sparsify the set of grasp choices by clustering antipodal hands based on distance and orientation. Grasp hypothesis that are nearby each other and that are roughly aligned in orientation are grouped together. Each cluster must be composed of a specified minimum number of constituent grasps. If a cluster is found, then we create a new grasp hypothesis positioned at the mean of the cluster and oriented with the "average" orientation of the constituent grasps. The next step is to select a grasp based on how easily it can be reached by the robot. First, we solve the inverse kinematics (IK) for each of the potential grasps and discard those for which no solution exists. The remaining grasps are ranked according to three criteria: (1) distance from joint limits (a piecewise function that is zero far from the arm joint limits and quadratic nearby the limits); (2) distance from hand joint limits (zero far from the limits and quadratic nearby limits); (3) workspace distance traveled by the hand starting from a fixed pre-grasp arm configuration. These three criteria are minimized in order of priority: first we select the set of grasps that minimize Criterion #1. Of those, we select those that minimize Criterion #2. Of those, we select the one that minimizes Criterion #3 as the grasp to be executed by the robot.

5.2 Objects Presented in Isolation

We performed a series of experiments to evaluate how well various parts of our algorithm perform in the context of grasping each of the 30 test set objects (Fig. 6). Each object was presented to the robot in isolation on a table in front of the robot. We characterize three variations on our algorithm:

1. **No Classification**: We assume that all hand hypotheses generated by the sampling algorithm (Algorithm 2) are antipodal and pass all hand samples directly to the grasp selection mechanism without classification as described in Sect. 5.1.
2. **Antipodal**: We classify hand hypotheses by evaluating the conditions of Definition 3 directly for each hand and pass the results to grasp selection.
3. **SVM**: We classify hand hypotheses using the SVM and pass the results to grasp selection. The system was trained using the 18-object training set as described in Sect. 4.4.

In all scenarios, a grasp trial was considered a success only when the robot successfully localized, grasped, lifted, and transported the object to a box on the side of the table. We evaluate *No Classification* and *SVM* for single-view and two-view registered points clouds over 214 grasps of the 30 test objects. Each object was placed in between 6 and 8 systematically different orientations relative to the robot.

Figure 7 shows the results. The results for *No Classification* are shown in columns *NC, 1V* and *NC, 2V*. Column *NC, 1V* shows that with a point cloud created using only one depth sensor, using the results of sampling with no additional classification results in an average grasp success rate of 58%. However, as shown in Column *NC, 2V*, it is possible to raise this success rate to 73% just by adding a second depth sensor and using the resulting two-view registered cloud. The fact that we obtain a grasp success rate as high as 73% here is surprising considering that the sample strategy employs rather simple geometric constraints. This suggests that even simple geometric constraints can improve grasp detection significantly. The results for *Antipodal* are shown in the column labeled *A, 2V*. We did not evaluate this variation for a one-view cloud because a two-view cloud is needed for Definition 3 to find any near antipodal hands. Compared to the other two approaches, *Antipodal* finds relatively few positives. This is because this method needs to "see" two sides of a potential grasp surface in order to verify the presence of a grasp. As a result, we were only able to evaluate this method over three poses per object instead of six or eight. In fact, *Antipodal* failed to find any grasps at all for four of the 30 objects.

Object	**number of poses**	**Succ. Rate A, 2V**		**number of poses**	**Success Rate**			
					NC, 1V	**NC, 2V**	**SVM, 1V**	**SVM, 2V**
Plush drill	3	100.00%		6	50.00%	66.67%	100.00	66.67%
Black pepper	3	100.00%		8	62.5%	62.50%	75.00	100.00%
Dremel engraver	3	100.00%		6	33.33%	50.00%	66.67	100.00%
Sand castle	3	100.00%		6	50.00%	33.33%	83.33	83.33%
Purple ball	0	NA		6	66.67%	100.00%	83.33	100.00%
White yarn roll	3	100.00%		8	87.50%	87.50%	87.50	75.00%
Odor protection	0	NA		8	50.00%	87.50%	87.50	75.00%
Neutrogena box	3	66.67%		8	25.00%	87.50%	87.50	87.50%
Plush screwdriver	3	100.00%		6	83.33%	87.50%	83.33	100.00%
Toy banana box	3	100.00%		8	100%	83.33%	87.50	75.00%
Rocket	3	100.00%		8	50.00%	87.50%	100.00	87.50%
Toy screw	3	100.00%		6	100.00%	100.00%	83.33	100.00%
Lamp	3	100.00%		8	62.50%	83.33%	87.50	87.50%
Toothpaste box	3	66.67%		8	87.50%	100.00%	87.50	87.50%
White squirt bottle	3	66.67%		8	25.00%	12.50%	75.00	87.50%
White rope	3	100.00%		6	66.67%	83.33%	83.33	100.00%
Whiteboard cleaner	3	100.00%		8	62.50%	75.00%	100.00	100.00%
Toy train	0	NA		8	87.50%	100.00%	87.50	100.00%
Vacuum part	3	100.00%		6	33.33%	66.67%	100.00	83.33%
Computer mouse	0	NA		6	33.33%	33.33%	66.67	83.33%
Vacuum brush	1	100%		6	50.00%	83.33%	66.67	50.00%
Lint roller	3	100.00%		8	75.00%	75.00%	87.50	100.00%
Ranch seasoning	3	100.00%		8	50.00%	75.00%	100.00	100.00%
Red pepper	3	100.00%		8	75.00%	75.00%	100.00	100.00%
Crystal light	3	100.00%		8	25.00%	37.50%	75.00	75.00%
Red thread	3	100.00%		8	75.00%	100.00%	100.00	100.00%
Kleenex	3	100.00%		6	33.33%	33.33%	83.33	83.33%
Lobster	3	66.67%		6	16.67%	83.33%	66.67	83.33%
Boat	3	100.00%		6	83.33%	100.00%	83.33	100.00%
Blue squirt bottle	2	100%		8	25.00%	50.00%	75.00	62.50%
Average		**94.67%**			**57.50%**	**72.92%**	**85.00%**	**87.78%**

Fig. 7 Single object experimental results. Algorithm variations are denoted as: A for antipodal grasps (see Sect. 4.1), NC for sampling without grasp classification (see Sect. 3), and SVM for our full detection system

Although *Antipodal* appears to be effective (94.7% grasp success rate), it is not very useful in practice since it works only for a small subset of possible object orientations. The results for *SVM* are shown in columns *SVM, 1V* and *SVM, 2V* (results for one-view and two-view point clouds, respectively). Interestingly, there is not much advantage here to adding a second depth camera: we achieve an 85.0% success rate with a one-view point cloud and an 87.8% success rate with a two-view registered cloud. Drilling down into these numbers, we find the following three major causes of grasp failure: (1) approximately 5.6% of the grasp failure rate in both scenarios is due to collisions between the gripper and the object caused by arm calibration errors (i.e. an inaccurate kinematic model that introduces error into the

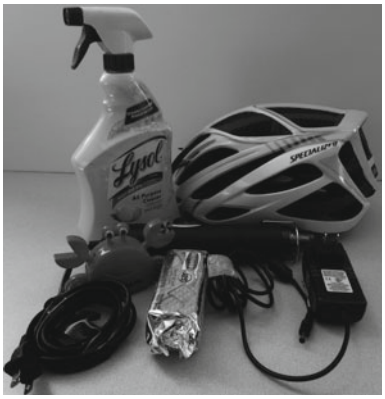


Fig. 8 Hard-to-see objects

calculation of forward/inverse kinematics) or collisions with observed or unobserved parts of the environment; (2) approximately 3.5% of the objects were dropped after a successful initial grasp; (3) approximately 2.3% of grasp failures in the two-view case (3.7% in the one view case) were caused by perceptual failures by our algorithm. The striking thing about the causes of failure listed above is that they are not all perceptual errors: if we want to improve beyond the 87.8% success rate, we need to improve performance in multiple areas (Fig. 8).

In the experiments described above, we eliminated seven objects from the test set because they were hard to see with our depth sensor (Asus Primesense) due to specularity, transparency, or color. We characterized grasp performance for these objects separately by grasping each of these objects in eight different poses (total of 56 grasps over all seven objects). Using *SVM*, we obtain a 66.7% grasp success rate using a single-view point cloud and a 83.3% grasp success rate when a two-view cloud is used. This result suggests: (1) our 87.8% success rate drops to 83% for hard-to-see objects; (2) creating a more complete point cloud by adding additional sensors is particularly important in non-ideal viewing conditions.

5.3 Objects Presented in Dense Clutter

We also characterized our algorithm in dense clutter as illustrated in Fig. 9. We created a test scenario where ten objects are piled together in a shallow box. We used exactly the same algorithm (i.e. *SVM*) in this experiment as in the isolated object experiments. We used a two-view registered point cloud in all cluttered scenarios. The 27 objects used in this experiment are a subset of the 30 objects used in the single object experiments. We eliminated the computer mouse and the engraver because they have cables attached to them that can get stuck in the clutter. We also removed the vacuum brush because the brush part cannot be grasped by the Baxter gripper in some configurations due to the 3–7 cm aperture limits. At the beginning of each run,

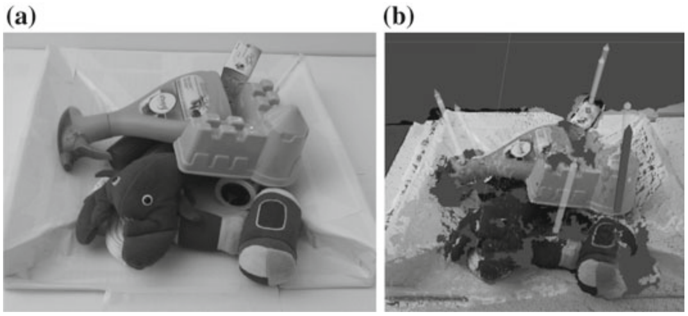


Fig. 9 Dense clutter scenario. **a** RGB image. **b** Output of our algorithm

we randomly selected 10 out of the 27 objects and placed them in a small rectangular container. We then shook the container to mix up the items and emptied it into the shallow box on top of the table. We excluded all runs where the sandcastle landed upside down because the Baxter gripper cannot grasp it in that configuration. A run was terminated when three consecutive localization failures occurred. In total, we performed 10 runs of this experiment.

Over all 10 runs of this experiment, the robot performed 113 grasps. On average, it succeeded in removing 85% of the objects from each box. The remaining objects were not grasped because the system failed to localize a grasp point three times in a row. Over all grasp attempts, 73% succeeded. The 27% failure rate breaks down into the following major failure modes: 3% due to kinematic modelling errors; 9% due to perceptual failures caused by our algorithm; 4% due to dropped objects following a successful grasp; and 4% due to collision with the environment. In comparison with the isolation results, these results have a significantly higher perceptual failure rate. We believe this is mainly due to the extensive occlusions in the clutter scenario.

6 Discussion

This paper proposes a new approach to localizing grasp poses of novel objects presented in clutter. The main contributions are: (1) a method of using grasp geometry to generate a set of hypotheses that focus grasp detection on relevant areas of the search space; (2) a method of using grasp geometry to label a training set automatically, thereby enabling the creation of a large training set grounded in grasp mechanics. As a result of these contributions, our method stands out from the existing work (for example [3, 4, 7, 11, 12]) in a couple of different ways. First, our method can detect 6-DOF hand *poses* from which a grasp is expected to succeed rather than detecting grasp *points* (typically 3-DOF) in a depth image or heightmap such as in [4, 12]. Second, our method of automatic training set generation should enable us to train better classifiers because we can generate as much training data as we want and reduce label noise because labels are generated based on objective mechanical conditions for a grasp. It should also be noted that our grasp hypothesis generation mechanism might be used independently of the grasp classification strategy. It is striking that the proposal mechanism alone (without any classification but with outlier removal) can yield a 73% grasp success rate when grasping objects presented in isolation. Essentially, this sample set constitutes a proposal distribution that should boost the performance of any classifier. Finally, the fact that we document a drop in grasp success rate from 87.8% for objects presented in isolation to 73% for objects presented in clutter suggests that clutter is a significantly more difficult problem that needs to be studied more closely.

Acknowledgements This work was supported in part by NASA under Grant No. NNX13AQ85G, ONR under Grant No. N000141410047, and the NSF under Grant No. 1427081.

References

1. Balasubramanian, R., Xu, L., Brook, P.D., Smith, J.R., Matsuoka, Y.: Human-guided grasp measures improve grasp robustness on physical robot. In: IEEE Intl Conf. on Robots and Automation, pp. 2294–2301. IEEE (2010)
2. Chang, L., Smith, J.R., Fox, D.: Interactive singulation of objects from a pile. In: IEEE International Conference on Robotics and Automation, pp. 3875–3882. IEEE (2012)
3. Detry, R., Ek, C.H., Madry, M., Kragic, D.: Learning a dictionary of prototypical grasp-predicting parts from grasping experience. In: IEEE International Conference on Robotics and Automation (2013)
4. Fischinger, D., Vincze, M.: Empty the basket - a shape based learning approach for grasping piles of unknown objects. In: IEEE International Conference on Intelligent Robot Systems (2012)
5. Fischinger, D., Vincze, M., Jiang, Y.: Learning grasps for unknown objects in cluttered scenes. In: IEEE International Conference on Robotics and Automation, pp. 609–616. IEEE (2013)
6. Glover, J., Popovic, S.: Bingham procrustean alignment for object detection in clutter. In: IEEE International Conference on Intelligent Robot Systems (2013)
7. Herzog, A., Pastor, P., Kalakrishnan, M., Righetti, L., Asfour, T., Schaal, S.: Template-based learning of grasp selection. In: IEEE International Conference on Robotics and Automation (2012)
8. Jiang, Y., Moseson, S., Saxena, A.: Efficient grasping from rgbd images: learning using a new rectangle representation. In: IEEE International Conference on Robotics and Automation (2011)
9. Katz, D., Kazemi, M., Bagnell, D., Stentz, A.: Clearing a pile of unknown objects using interactive perception. In: IEEE International Conference on Robotics and Automation (2013)
10. Klingbeil, E., Rao, D., Carpenter, B., Ganapathi, B., Ng, A., Khatib, O.: Grasping with application to an autonomous checkout robot. In: IEEE International Conference on Robotics and Automation (2011)
11. Kroemer, O., Ugur, E., Oztop, E., Peters, J.: A kernel-based approach to direct action perception. In: IEEE International Conference on Robots and Automation, pp. 2605–2610. IEEE (2012)
12. Lenz, I., Lee, H., Saxena, A.: Deep learning for detecting robotic grasps. In: Robotics: Science and Systems (2013)
13. Murray, R.M., Li, Z., Sastry, S.S., Sastry, S.S.: A Mathematical Introduction to Robotic Manipulation. CRC press, Boca Raton (1994)

14. Nguyen, V.: Constructing force-closure grasps. IEEE Int. Conf. Robot. Autom. **3**, 1368–1373 (1986)
15. Rusu, R., Blodow, N., Beetz, M.: Fast point feature histograms (fpfh) for 3d registration. In: IEEE International Conference on Robots and Automation (2009)
16. Saxena, A., Driemeyer, J., Ng, A.: Robotic grasping of novel objects using vision. Int. J. Robot. Res. **27**(4), 157 (2008)
17. Sudsang, A., Ponce, J.: New techniques for computing four-finger force-closure grasps of polyhedral objects. IEEE Int. Conf. Robot. Autom. **2**, 1355–1360 (1995)
18. Taubin, G.: Estimation of planar curves, surfaces and nonplanar space curves defined by implicit equations, with applications to edge and range image segmentation. IEEE Trans. PAMI **13**, 1115–1138 (1991)
19. ten Pas, A., Platt, R.: Localizing handle-like grasp affordances in 3d point clouds. In: International Symposium on Experimental Robotics (2014)
20. Tombari, F., Salti, S., Di Stefano, L.: Unique signatures of histograms for local surface description. In: Computer Vision–ECCV 2010, pp. 356–369. Springer (2010)

Grasping with Your Brain: A Brain-Computer Interface for Fast Grasp Selection

Robert Ying, Jonathan Weisz and Peter K. Allen

1 Introduction

People with restricted mobility currently require significant infrastructural support in order to perform activities of daily living (ADL), including things like manipulating objects, opening doors, and other basic actions that able-bodied people often take for granted. With the current state-of-the-art robotic arms, hands, and perception systems, it is clear that robotic grasping systems could help reduce the dependency severely disabled individuals have on live-in caretakers, and provide them with the ability to actively interact with their environment.

However, the robotic grasping systems which show the greatest promise in performing ADL tend to be highly complex, involving high degree of freedom manipulators and precise control to achieve their objectives. It is therefore important to present users with a high-level interface to these grasping systems that can operate with relatively little training.

In previous work [21–23], we have presented a shared-control online grasp planner that collaboratively determines feasible grasps under the active direction of a user through a low-bandwidth interface. We have demonstrated the efficacy of this system using a variety of facial EMG-based devices in moderately cluttered scenes. However, this interface depends on the ability of the user to trigger relevant facial muscles repeatably and reliably.

R. Ying · J. Weisz · P.K. Allen (✉)
Department of Computer Science, Columbia University, 500 W. 120th Street, M.C. 0401, New York, NY 10027, USA
e-mail: allen@cs.columbia.edu

R. Ying
e-mail: robert.ying@columbia.edu

J. Weisz
e-mail: jweisz@cs.columbia.edu

A. Bicchi and W. Burgard (eds.), *Robotics Research*, Springer Proceedings in Advanced Robotics 2, DOI 10.1007/978-3-319-51532-8_20

In this work, we extend this system to an EEG-based system, which has a number of advantages. Firstly, the neurological phenomena used in the system is a subconscious reaction to visual stimuli, and therefore needs very little relevant user expertise to operate. Secondly, the planner can take advantage of visual ambiguity between functionally similar grasps to achieve fast convergence in the shared-control paradigm. The user acts as a filter for the planner, directing it to a desired approach direction and filtering proposed candidates until a reasonable one is found. Three users were able to use this system with minimal training to pick up a variety of objects in a semi-cluttered scene.

2 Prior Work

Brain-Computer Interface (BCI) control over prosthetic and assistive manipulators has been the subject of a great deal of research, through many different strategies and input modalities. Recently there has been a resurgence of interest in this field. One widely cited recent advance was reported by Vogel et al. [19], who showed that a subject with a BrainGate cortically-implanted electrode can use a robotic manipulator to retrieve a drink container by controlling the end-effector location and the opening and closing of the hand. However, this approach requires an invasive device capable of recording a large number of high quality signals.

Noninvasive EEG systems have been demonstrated in a number of simpler tasks. In [15], surface electrode signals related to eye gaze direction are used to control 2D arm position and EEG signals are used to detect eye blinks to control gripper closing. In [9], hand opening/closing and elbow flexion/extension are controlled by EEG signals.

The majority of previous work using EEG control concentrates on trajectory control. However, it has been shown that users find BCI control easier using even higher-level, goal-oriented paradigms [16]. We have begun to see work that attempts to exploit higher-level abstractions to allow users to perform more complex tasks with robotic arms. In [2], EEG signals were used to select targets for pick and place operations for a small humanoid robot. In [20], the authors used EEG signals to control pick and place operations of a 4-DOF Staubli robot. Bryan et al. [3] presented preliminary work in extending this approach to a grasping pipeline on the PR2 robot. In that work, a 3D perception pipeline is used to find and identify target objects for grasping and EEG signals are used to choose between them.

Recently, some authors including [11, 12] have explored shared control paradigms which integrate computer vision, intra-cortical EEG recording, and online planning to perform reaching, grasping, and manipulation tasks. These works are promising, but rely on the higher fidelity control available from implanted devices. In [4], the planner presented in our work, which is focused on acquiring an appropriate grasp of the object with arbitrarily complex hands, was integrated with a similar system. In this work, we introduce an interface to the user which allows them to control higher level, more abstract goals with lower throughput devices, which could be made complimentary to these other shared controlled paradigms.

3 Methods

3.1 Overview

We present here a prototype of an assistive grasping system which integrates a BCI driven user interface with a perception pipeline, a lightweight mountable manipulator, in this case the 6-DOF Mico arm with a two-finger underactuated gripper [10], and an online grasp planning system to allow a user to grasp an object in moderately cluttered scenes. It decomposes the grasping task into a multi-step pipeline where each step generates a visual representation of the options the user can take. Some options which cannot be visually represented, such as returning to a previous state, are presented as white text on a black background. At each stage, the online planning system derives a set of reasonable possible actions and presents them to the user, reducing the complex task of grasping an object in cluttered scenes to a series of decision points that can be navigated with a low throughput, noisy input such as an EEG headcap. Figure 1 shows a healthy subject in our validation study using the system to grasp a bottle of laundry detergent in a typical scene.

3.2 Grasp Planning

This system uses the Online Eigengrasp Planner introduced by Ciocarlie et al. in [5]. This planner uses simulated annealing to generate grasp candidates by projecting desired contacts points on to the target object to find grasps likely to result in a force closed grasp. In order to make this task computationally tractable, a reduced subspace of the hand's full configuration is sampled. In the case of the a simple gripper such as that on the Mico, this may not be necessary, but the use of this planner makes the computational cost of using a more complex hand nearly the same as this simpler hand. Candidate grasps in near contact positions are refined to completed grasps by kinematic simulation of closing the hand at a predefined set of joint velocities.

The resulting contacts are ranked by the maximum wrench perturbation force they are capable of resisting, as described in [6], and the closeness of the alignment between the hand and the object's surface. If the quality metric is above 0.2 and all of the dot products of the normal direction of the hand and object is above 0.8 for all of the contact points, the grasping pose is tested for reachability using the PRM planner of *MoveIt!* [18]. When the scene is cluttered, the motion planner for the reaching motion is slow and likely to fail. In order to make this problem more computationally tractable, we cache previous solutions as grasps are planned.

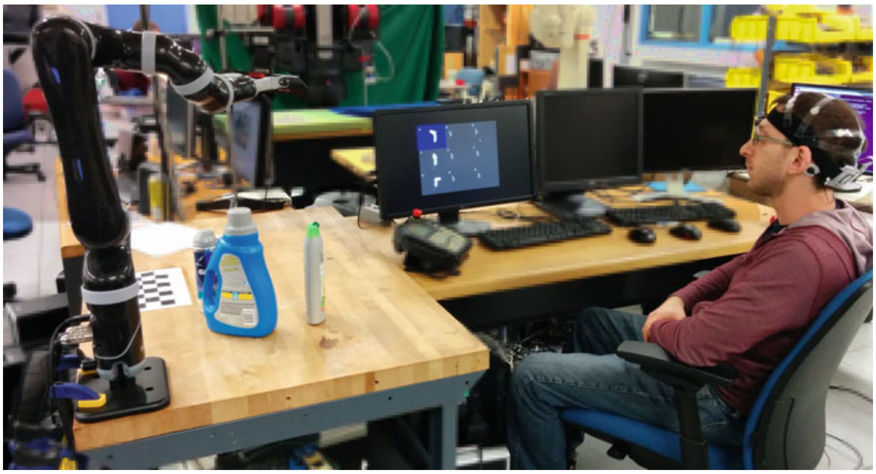


Fig. 1 The subject guiding the system through the active refinement phase. On the *left side* is the robotic manipulator and three containers in the grasping scene. On the *right* is a subject using the system through the B-Alert EEG cap, which is relatively unobtrusive and can be worn for long periods of time. The options for the active refinement stage are presented on the monitor in front of the subject in a grid to allow the subject to pick the one they intend to select during the serial presentation. In this example, at least one of the grasps found in the database for the object was reachable, and is highlighted in *blue* in the *upper left corner* of the grid. The user may choose to execute the highlighted grasp, or to re-seed the planner with one of the other nine grasps and then re-enter the active refinement phase with a new highlighted grasp

Whenever a previous solution ends near the new candidate grasp pose, we plan from its end point to the new grasp candidate. Since the nature of our grasp planner produces many nearby solutions, this makes the reachability filter significantly faster and more robust. Grasps are ranked first by reachability, then by the grasp quality, and finally by the maximal surface misalignment.

The neighbor generating function of the simulated annealing planner is biased towards a configuration demonstrated by the user. By controlling this seed configuration, the user controls the resulting set of candidates that will presented to them. This allows the user to find a grasp for a particular purpose by iteratively picking the grasp whose pose is nearest to the grasp that they are looking for.

3.3 One-of-Many Selection

The EEG interface presented in this paper is based on an "interest" detector which can be used to provide a one-of-many selection between various options. This "interest" signal paradigm is based on the work in [14]. The options are presented as a stream of images, and the subject is primed to look for particular images that suit some criterion. This paradigm is known as *Rapid Serial Visual Presentation* (RSVP). Spikes in

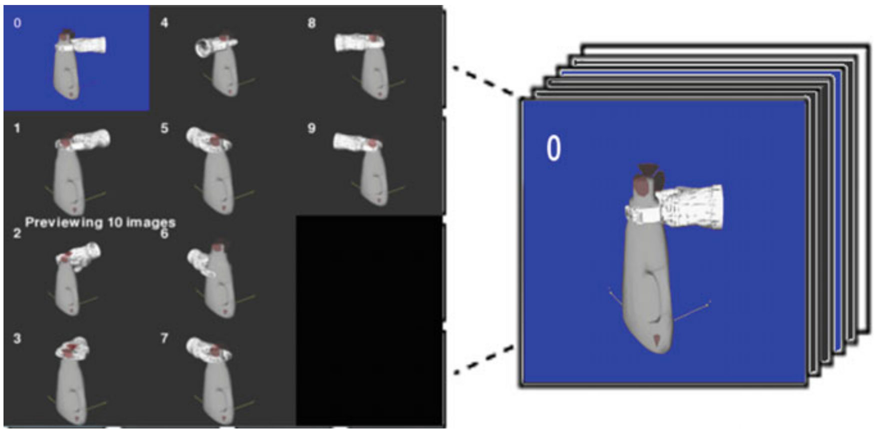


Fig. 2 The grasp planning system compiles a set of images representing potential actions, for example a set of grasps as seen in this image. The image options are tiled together to form the summary pane seen on the *left*, which lets the user pick out the one that reflects their desire. The images are then shuffled, with repetitions, into a stream that is serially presented to the user as described in Sect. 3.3.3

EEG activity which correlate with "interest" are connected with the image that was presented at the time the EEG activity was evoked, which is then used to derive the user's desired input.

Previous work with this paradigm has asked the subject to look for objects of a particular category. In our system, the images represent actions that are suggested by the grasp planner, which the subject may not have had previous experience with. In this case, the subject must be given time to analyze the options and primed to find the features which make their desired option visually distinct from similar options. In Fig. 2, we illustrate the summary pane containing a grid of all of the options, which are then shuffled and presented to the user. In Fig. 1, you can see the subject reviewing the options in a summary pane before the serial presentation of them begins.

One major advantage of this paradigm is that it generalizes a single interaction across all phases of the grasp planning pipeline. The system only needs to be trained to recognize the "interest" signal for each subject. Afterwards, the subject's interaction with each phase is the same, and the system does not require phase-specific training.

3.3.1 EEG Input

Our current implementation uses a B-Alert X10 EEG system from Advanced Brain Monitoring (Carlsbad, CA), which provides 9 electrodes positioned according to the 10–20 system and a pair of reference channels. The EEG data is acquired at 256 Hz, with 60 Hz notch and 0.5 Hz high-pass filters applied before any additional processing. The EEG interest metric is based on that described in [8, 14, 17], with some additional normalization and post-processing.

More information on this system can be found online at the manufacturer's website [1]. As can be seen in Fig. 1, the cap and device are relatively minimalistic, and can be comfortably worn for an hour at a time without requiring rewetting or reseating of the electrodes. With the advent of the OpenBCI project [13] and similar efforts towards low cost, open hardware EEG devices, a low cost solution with similar capabilities may be on the horizon.

3.3.2 EEG Interest Metric

The EEG interest metric is based on the one used in [8, 14, 17]. In essence, it assumes that the P300 signal resulting from a particular image varies with a resolution of 100 ms. For each block, it examines the time period from 100 to 1200 ms after the input stimulus as separate 100 ms blocks, combined in a linear model:

$$y_{sn} = \sum_i w_i x_{in} \qquad y = \sum_n v_n y_{sn} \tag{1}$$

where each x_{in} is the reading at a specific electrode i at some time period n, y_{sn} is the weighted total score over a single 100 ms block and y is the combined score for the 1100 ms time period following the stimulus. The weights w_i are learned from the training data so as to maximize the difference between target and non-target images in each time block using Fisher linear discriminant analysis [7]. Then, the weights v_n are determined by applying logistic regression on the training data.

In training, we additionally compute summary statistics for both target and non-target images, which are used later to normalize the individual readings per trial.

3.3.3 Option Generation

To generate the RSVP sequence, the system randomly selects each option to appear between three and seven times. The sequence is then randomly shuffled, with the constraint that the same option does not appear in two consecutive image presentations. This method has, in experimental data, been sufficient to trigger the "oddball" response that is necessary for the P300 signal.

If there are less than five options, the system will automatically fill in distractor image options to make this constraint more feasible. The images are dependent on the phase and attempt to minimize the visual difference between the distractor and the original, so as to avoid unintentionally triggering the P300 signal. For example, in the object selection state, the distractor options are of the scene with no objects selected; whereas in the active refinement state they are images of the object with no visible grasp.

More formally, the grasp planner generates a set of options $Q = T \cup G$, where T is a set of strings representing textual options (e.g. “Rerun object recognition”), and G is a set of potential grasp or object images. If $|Q| < 5$, the selection system then adds $5 - |Q|$ distractor images d to result in Q'.

From Q', we generate the sequence of images I as follows:

$$I = \text{shuffle}(I_{q_1 1}, I_{q_1 2}, \ldots, I_{q_{1 c_1}}, I_{q_2 1}, \ldots, I_{q_{k c_k}}) \tag{2}$$

where $k = |Q'|$, $q_i \in Q'$ and $c_j \sim U(3, 7)\ \forall\ j \in [1, k]$.

The images are each presented at 4 Hz, and preliminary EEG scores e_i are assigned. We then aggregate each of the $n = \sum_{j=1}^{k} c_j$ images by their option, and determine whether or not the user has made a selection.

To test if the user has consciously selected any of the images, we sort the images by their EEG scores, and then split it into a group of size x and $n - x$. We vary x so as to maximize the change in the average measured EEG score:

$$x^* = \arg\max_{x \in [1,n]} \left(\frac{1}{n} \sum_{i=1}^{n} e_i - \frac{1}{n-x} \sum_{i=x+1}^{n} e_i \right) \tag{3}$$

If $x^* > \max(0.2n, 7)$, we determine that the user had not made a choice. In practice, this is a highly reliable means of checking whether the user was paying attention and attempting to make a selection.

If $x^* \leq \max(0.2n, 7)$, we compute a smoothed similarity score using the top x^* positions.

3.3.4 Option Scoring

The options are scored using a smoothed similarity metric, represented as a symmetric matrix $S \in R_{k \times k}$, computed such that $S_{ii} = 1$ and $S_{ij} = S_{ji} \in [-1.0, 1.0]$.

We can then construct the weighted score vector W as

$$W_q = \sum_{i=0}^{x^*} S_{q_i q} \tag{4}$$

where q_i is the option corresponding to I_i, and return

$$q^* = \arg\max_{q \in Q'} W_q \tag{5}$$

This scoring method introduces a bias towards groups of similar options, and in essence allows a near-miss selection to nonetheless help select the desired option. From our experiments, this is particularly helpful with subjects who are less experienced with the system, as they often make minor mistakes during the selection process.

We note that this is equivalent to a simple voting scheme if $S = I_{k \times k}$, i.e. the identity matrix of size k. Thus, for options where there is no obvious similarity metric, such as textual or distractor options, we use the corresponding rows and columns from the identity as default.

3.4 Grasping Pipeline

There are four states that the user progresses through when attempting to formulate a grasp, *Object Selection*, *Grasp Selection*, *Grasp Refinement*, and *Confirmation*. The pathway is illustrated in Fig. 3.

3.4.1 Object Selection State

In this stage, an object recognition system is used to retrieve models from a database that fit the scene. An image representing selection of each object is generated as shown in the "summary pane" in Fig. 4a, with the target object highlighted green in each potential selection. Between the various images only the highlighted object changes. An additional state is presented that allows the user to run the recognition system again. If fewer than eight objects are detected, additional distractor images of the scene with no highlighted object are generated to act as distractor images which help establish the background level of EEG activity. The user is instructed to just look for the object they want to grasp as the image stream is shown. In this state, the similarity matrix is the identity matrix over the viable options, as the objects are highly dissimilar.

3.4.2 Grasp Selection and Refinement State

Once the object is selected, the system moves into the grasp selection state. The user's interaction with the grasp selection state and refinement states are very similar. Examples of the "summary pane" for these phases are shown in Fig. 4b, c.

In the grasp selection state, the set of preplanned grasps is retrieved and placed in an arbitrary order. Each of the grasps is visually distinct, and supplies the planner with an approach angle to start with. One additional text option is presented, which sends the user back to the object selection stage. When any grasp is detected as a valid selection, the system enters the grasp refinement state, setting the seed grasp g_s to the one just selected. If fewer than eight grasps are available, images of the object without a visible grasp are used as distractors.

In the grasp refinement state, the online planner begins populating the grasp list with more grasps that are similar to g_s. After allowing the planner to run for fifteen

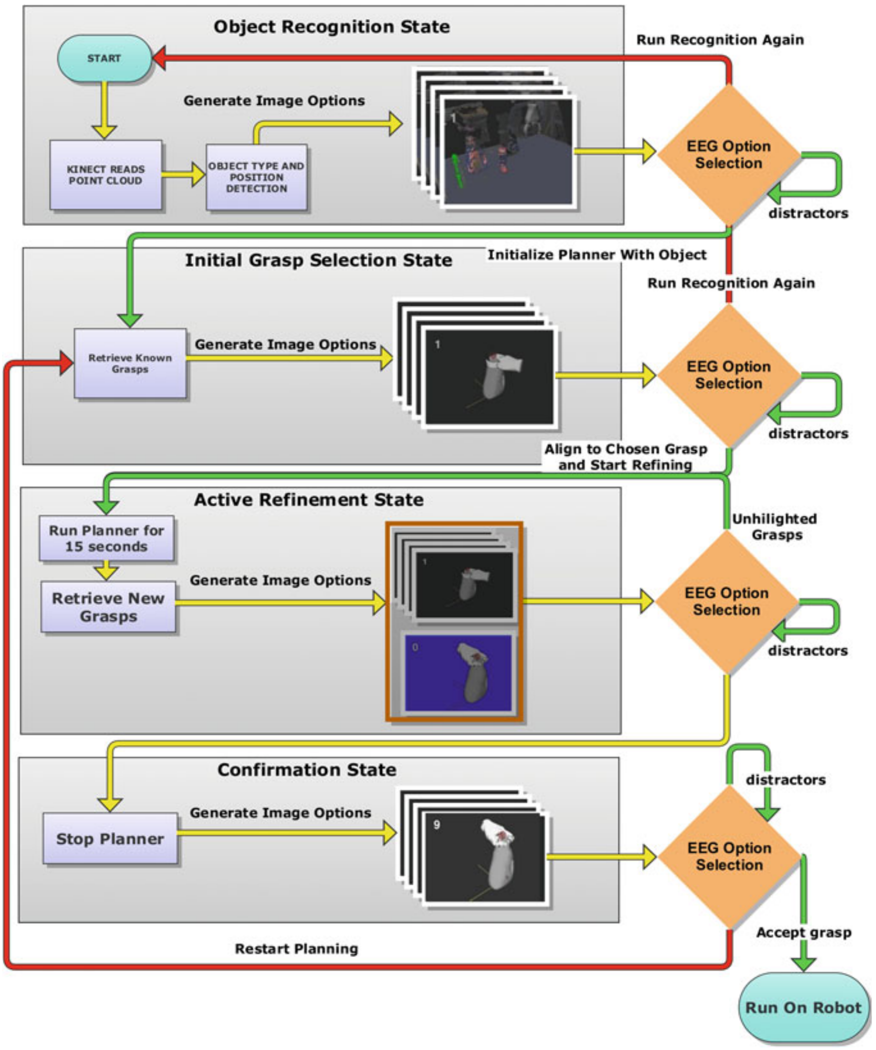
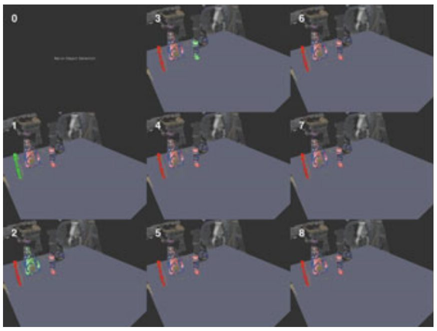
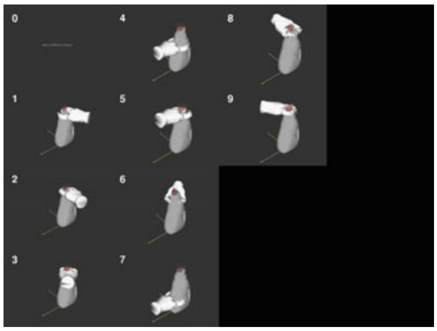
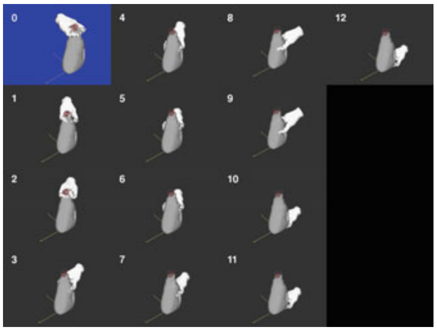


Fig. 3 A diagram outlining the EEG RSVP-driven grasping pipeline. In each phase, a series of images is generated representing the available options, as described in Sect. 3.4. A summary pane of the image options generated at each phase is presented in more detail in Fig. 4a–d

seconds, the available grasps are presented to the user. In most cases, this will be a list of at least ten potential grasps. As each grasp is generated, it is checked for reachability– while even non-reachable grasps are sent to the user, the grasp refinement state cannot be exited until a reachable grasp has been selected. The highest quality reachable grasp g_p is highlighted in blue in the user interface (Fig. 4c), so that the user has feedback as to what the planner is deciding between.


(a) Object Selection State


(b) Grasp Selection State


(c) Grasp Refinement State

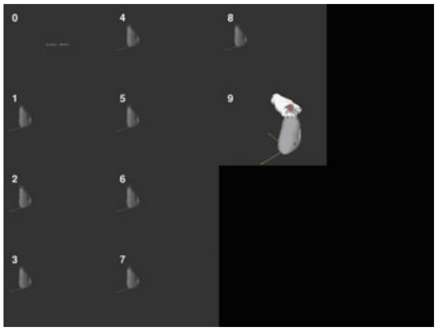

(d) Confirmation State

Fig. 4 **a** The three objects visible in the planning screen are presented to the user, with the object to be selected highlighted in *green*. The scene is overlaid with the pointcloud data from the Kinect, so that the user can verify that the objects have been recognized and positioned correctly within the scene. **b** The initial set of pre-computed grasps from a database. The user may choose to go back to the previous phase and choose a different object, or to seed the online grasp planner using one of the available grasps. **c** An updated set of grasps in the active refinement state, generated from grasp number 8 from Fig. 4b. The selected grasp g_s is reachable and highlighted in *blue*. Note that the generated grasps are visually distinct, but still have small groups of functionally identical grasps. **d** There are effectively only two options in the confirmation state, which acts as a final check to determine whether the selected grasp g_s is the one that the user would like to execute

Once the user selects a grasp, the planner updates g_s to the new grasp's hand state and approach vector. If the updated $g_s = g_p$, then the user exits grasp refinement and enters the confirmation state.

In this phase, we also take advantage of visual ambiguity. We compute the similarity matrix S as follows:

$$S_{ij} = \langle \hat{u}_i, \hat{u}_j \rangle = \cos\theta_{ij} \tag{6}$$

where $\hat{u}_i$ and $\hat{u}_j$ are the approach directions for the grasps under consideration.

As per usual, for distractor images and text, we set all of the rows and columns representing non-grasp options to the default identity matrix.

This reduces the number of ambiguous selections that occur when the user has no strong preference among a subset of very similar grasps, which is a fairly common outcome under experimental conditions.

Furthermore, within any 20° cone, we allow only five grasps to be added to the option set. When more than five are found, the oldest grasp (least-recently-generated) is penalized and moved to the end of the list, behind grasps with a more different approach direction. This forces some heterogeneity to remain in the grasps that are presented to the user, while additionally allowing the user to "walk" the grasping point and direction to a new approach direction, even if it is not presented in any of the previous options.

In the confirmation state, the user is shown an image corresponding to the selected grasp g_s, along with a set of distractor images presenting just the target object without a grasp. The user also has the text option "Go Back-Replan", which returns the user to the object selection phase. Selecting the grasp again confirms the grasp for execution and sends the desired grasp to the robot.

3.4.3 Execution State

In the execution state, the user is presented with only three text options: "Restart Execution", which restarts the execution if it has failed, telling the robot to return to its home position and attempt to grasp again; "Stop Execution", which stops the robot from continuing the execution and returns to the confirmation state, and a set of distractor images which say "Distractor Image."

4 Experiment

We have validated this system on three subjects, asking them to lift each of three objects visible in Fig. 5: a shaving gel bottle, a detergent bottle, and a shampoo bottle. The three objects are placed arbitrarily within the field of view of the Kinect camera, such that they do not fully occlude each other. In each case, we have verified that the object is within the reachable working area of the Mico arm, so there is at least one feasible grasp.

The subject is given the opportunity to inspect the scene, and is then asked to lift each of the objects three times from either the top, from the side, or at their own discretion. All testing was approved by the Institutional Review Board of Columbia University under Protocol IRB-AAAJ6951.

The experimental setup can be seen in Fig. 1, and a video of a subject going through the pipeline is available online.[1]

[1] http://isrrvideo.wc.aeturnalus.com/.

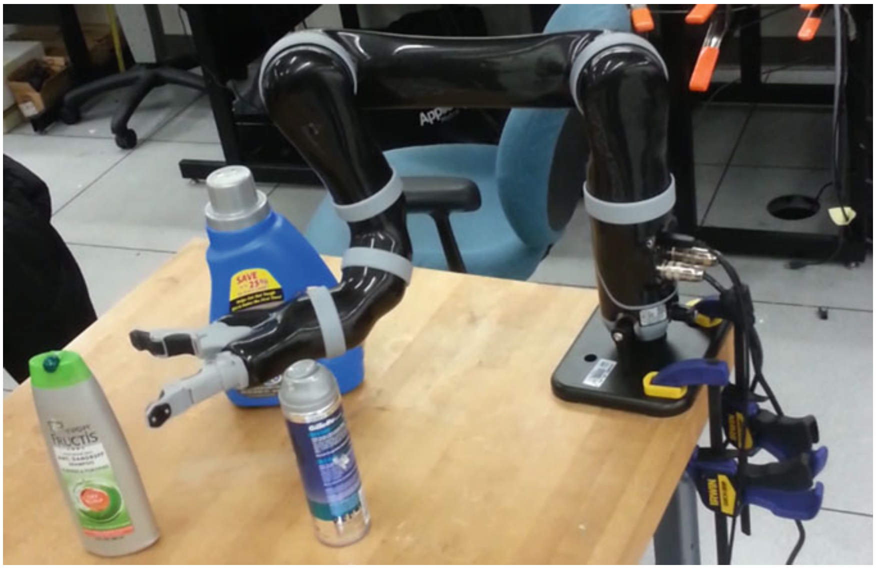


Fig. 5 A typical grasp of the shampoo bottle from the side in the cluttered scene. Note that the hand is just able to fit between the other objects to grasp the desired target. Note that the ability to plan this grasp in such a restricted environment is an indication that this system is very successful at handling the cluttered scene

4.1 Training

To demonstrate the various stages in the pipeline, the user is shown the system running under keyboard control, where each option can be selected by pressing its corresponding key. We allow the subject to walk through the stages of the grasping procedure as many times as they ask, (always less than five), while explaining what is being visualized at each step.

The EEG classifier weights described in Eq. 1 must be retrained each time the headset is placed on the user's head. This process takes approximately ten to fifteen minutes, and also serves to help familiarize the user with the RSVP paradigm.

During the training phase, the user is shown a "block" of 42 images. 40 of these images are selected uniformly from a set of fifteen object models similar to those presented during the object selection phase, while the remaining two are marked as "target" images and selected from a set of four images of bowls. Unlike the object selection phase, however, these object images are presented one at a time, without the other objects visible. There is also no "summary pane", as the images would be too small to practically see. Instead, the subject is shown the set of four potential target images.

In each block, the subject is told that there will be exactly two target images, and is asked to search for them in the sequence. After a block of images has been

Table 1 Experimental results from three subjects

Grasp	Subject	Misselections	Refinement iterations	Time(s)
Detergent bottle top	1	0	1	120
	2	2	3	150
	3	1	2	135
Detergent bottle side	1	0	1	120
	2	1	2	135
	3	0	1	120
Detergent bottle choice	1	0	10	270
	2	0	2	135
	3	3	5	180
Shampoo bottle top	1	0	1	135
	2	0	1	120
	3	0	1	150
Shampoo bottle side	1	0	1	120
	2	1	1	135
	3	0	2	135
Shampoo bottle choice	1	1	1	210
	2	1	3	120
	3	0	1	150
Shaving gel top	1	0	2	180
	2	1	1	120
	3	0	2	135
Shaving gel side	1	1	2	135
	2	0	1	120
	3	0	2	150
Shaving gel choice	1	0	2	120
	2	0	1	120
	3	0	2	180

presented, the user is also shown the location of the two target images in the sequence and, separately, where the classifier placed those images in a list sorted by detected interest level.

The user is presented blocks at a self-paced rate until at least 20 blocks have been presented, and the user is able to consistently place the two target images in the top three sort positions. The latter condition is usually fulfilled before the former.

4.2 Results

The results of this experiment are summarized in Table 1. In all cases, the subjects were successful in selecting reasonable grasps that lifted the object. However, the system did not always detect the option that the subject wanted correctly. In the table, the third column describes the number of misselections, which represents the number of times that the user inadvertently selected an option. Because the pipeline allows stepping back through each phase, this is not fatal, though it does result in a longer task duration (shown in the fifth column in fifteen-second increments). Detected selections of known "distractor" images are not considered misselections, as they do not elicit any actual action that changes the state of the system. The fourth column describes the number of iterations of the "grasp refinement" stage the user stepped through in order to find an acceptable grasp.

When grasping the detergent bottle, Subject 1 chose to attempt to grasp the handle of the object starting from a top grasp, eliciting the "walking" behavior described in Sect. 3.4.2. This necessitated a correspondingly large number of iterations of the refinement state.

The largest number of misselections came from the users accidentally selecting the option to rerun object detection in the "object selection" phase. Misselections of the wrong grasp during the "grasp refinement" stage when the user actually wanted to accept the current best grasp (g_s, above) and continue to the confirmation state also occurred, but these mistakes were quickly recoverable because similar grasps were very likely to be present in the next set of presented grasps.

5 Conclusions

These results are encouraging, and demonstrate that a relatively fast and effective pipeline based off of only EEG data is workable. The experiment revealed some issues, specifically in terms of how images are generated when representing abstract concepts (e.g. text-based image options, and the "selected" grasp in the "grasp refinement" stage).

The most common misselection was the command to redo the object detection during the "object selection" phase. This is probably because the difference between the images representing object selections and the text option image is large and somewhat startling, which elicits a reaction from the subject. A similar issue was seen in the initial attempt to use the system with Subject 2, who selected the blue image option every time it was presented in the "grasp refinement" stage, until the brightness of the background was reduced by 50%. After this modification, the subject had no trouble making the correct selection in the "grasp refinement" stage. Subject 2 sometimes had trouble making the correct selection in the "confirmation" stage, possibly because the distractor images were too similar, which makes the task too different from the one that the classification system is trained on.

There is an enormous space of design parameters that can be explored to potentially resolve some of these issues to produce a more robust system. One option would be adapting the thresholds used by the classifier based on the content of the image options. Another option would be to modify the training set of images to be more similar to the images presented in the task stage. While the current training regime has proven to be somewhat generalizable, it may not be adequately representative of the responses that are elicited by large stimuli like the blue background of the "selected image." Finally, some calibration procedure for modifying the images based on the responses they elicit from the user may need to be incorporated into the training regime.

The extension of the online Human-in-the-Loop planner to this EEG based image streaming paradigm has just begun. In its current implementation, the subject decisions are elicited at fixed points of the pipeline. Future work will move towards attempting to integrate the EEG data in a more real-time strategy, perhaps being fully embedded into the augmented reality environment. Although this system is primarily designed as a component of an assistive robotic manipulation platform, the real time system would be useful even for able-bodied users as a fast, passive filter for eliciting feedback from the user.

Finally, another approach to be explored in the future may be to combine this method with the sEMG method from [21]. This multimodal strategy would incorporate an EEG-based classifier for directing the planner towards the user's preferences while a facial EMG input is used to signal discrete decisions. Such a system would address the shortcomings of each individual modality – allowing the system to quickly filter reasonable options, where accuracy may be less important so long as it is somewhat conservative, while making the final selections, which affect the state of the robot and may result in inappropriate or potentially damaging behaviors, more robust.

Acknowledgements The authors would like to gratefully acknowledge the help of Dr. Paul Sajda and Dr. David Jangraw in this work. This work was supported by NSF Grant IIS-1208153.

References

1. Advanced Brain Monitoring. http://www.advancedbrainmonitoring.com/xseries/x10/
2. Bell, C.J., Shenoy, P., Chalodhorn, R., Rao, R.P.: Control of a humanoid robot by a noninvasive brain-computer interface in humans. J. Neural Eng. **5**(2), 214 (2008)
3. Bryan, J., Thomas, V., Nicoll, G., Chang, L., Rao, R.: What you think is what you get: brain-controlled interfacing for the pr2. In: IROS (2011)

4. Ciocarlie, M., Clanton, S., Spalding, M., Allen, P.: Biomimetic grasp planning for cortical control of a robotic hand. In: Proceedings of IROS, pp. 2271–2276 (2008)
5. Ciocarlie, M.T., Allen, P.K.: Hand posture subspaces for dexterous robotic grasping. Int. J. Robot. Res. **28**(7), 851–867 (2009)
6. Ferrari, C., Canny, J.: Planning optimal grasps. In: Proceedings of the International Conference on Robotics and Automation (1992)
7. Fisher, R.A.: The use of multiple measurements in taxonomic problems. Annals of Eugenics **7**(2), 179–188 (1936)
8. Gerson, A., Parra, L., Sajda, P.: Cortically coupled computer vision for rapid image search. IEEE Trans. Neural Syst. Rehabil. Eng. **14**(2), 174–179 (2006)
9. Horki, P., Solis-Escalante, T., Neuper, C., Müller-Putz, G.: Combined motor imagery and ssvep based bci control of a 2 dof artificial upper limb. Med. Biol. Eng. Comput. **49**(5), 567–577 (2011)
10. Kinova Robotics Mico. http://kinovarobotics.com/products/mico-robotics/
11. Lampe, T., Fiederer, L.D., Voelker, M., Knorr, A., Riedmiller, M., Ball, T.: A brain-computer interface for high-level remote control of an autonomous, reinforcement-learning-based robotic system for reaching and grasping. In: Proceedings of International Conference on Intelligent User Interfaces, IUI ’14 (2014)
12. Muelling, K., Venkatraman, A., Valois, J.S., Downey, J., Weiss, J., Javdani, S., Hebert, M., Schwartz, A.B., Collinger, J.L., Bagnell, J.A.: Autonomy infused teleoperation with application to bci manipulation. arXiv preprint arXiv:1503.05451 (2015)
13. OpenBCI. http://www.openbci.com
14. Pohlmeyer, E.A., Wang, J., Jangraw, D.C., Lou, B., Chang, S.F., Sajda, P.: Closing the loop in cortically-coupled computer vision: a brain-computer interface for searching image databases. J. Neural Eng. **8**(3), 036025 (2011)
15. Postelnicu, C.C., Talaba, D., Toma, M.I.: Controlling a robotic arm by brainwaves and eye movement. In: Technological Innovation for Sustainability. Springer (2011)
16. Royer, A.S., Rose, M.L., He, B.: Goal selection versus process control while learning to use a brain–computer interface. J. Neural Eng. **8**(3), 036,012 (2011)
17. Sajda, P., Pohlmeyer, E., Wang, J., Parra, L., Christoforou, C., Dmochowski, J., Hanna, B., Bahlmann, C., Singh, M., Chang, S.F.: In a blink of an eye and a switch of a transistor: cortically coupled computer vision. Proc. IEEE **98**(3), 462–478 (2010)
18. Sucan, I.A., Chitta, S.M.: (2013). http://moveit.ros.org
19. Vogel, J., Haddadin, S., Simeral, J.D., Stavisky, S.D., Bacher, D., Hochberg, L.R., Donoghue, J.P., van der Smagt, P.: Continuous control of the dlr light-weight robot iii by a human with tetraplegia using the braingate2 neural interface system. In: Experimental Robotics, pp. 125–136. Springer (2014)
20. Waytowich, N., Henderson, A., Krusienski, D., Cox, D.: Robot application of a brain computer interface to staubli tx40 robots-early stages. In: World Automation Congress. IEEE (2010)
21. Weisz, J., Elvezio, C., Allen, P.K.: A user interface for assistive grasping. In: 2013 IEEE/RSJ International Conference on Intelligent Robots and Systems (2013)
22. Weisz, J., Shababo, B., Dong, L., Allen, P.K.: Grasping with your face. Springer Tracts in Advanced Robotics pp. 435–448 (2013)
23. Weisz, J., Barszap, A.G., Joshi, S.S., Allen, P.K.: Single muscle site semg interface for assistive grasping. IROS (2014)

Spine Balancing Strategy Using Muscle ZMP on Musculoskeletal Humanoid Kenshiro

Yuki Asano, Soichi Ookubo, Toyotaka Kozuki, Takuma Shirai, Kohei Kimura, Shunichi Nozawa, Youhei Kakiuchi, Kei Okada and Masayuki Inaba

1 Introduction

Several studies have been conducted on the development of a stabilizer for humanoids, as this is a very important component for humanoids that are unstable under bipedal conditions. Some of these studies focus on posture stabilization during bipedal walking [2, 9]. Others concentrate on the development of balancing strategies with force control [4, 11]. In such stabilizer studies, the concept of Zero Moment Point(ZMP)

Y. Asano (✉) · S. Ookubo · T. Kozuki · T. Shirai · K. Kimura · S. Nozawa · Y. Kakiuchi · K. Okada · M. Inaba
Department of Mechano-Informatics, Graduate School of Information Science and Technology, The University of Tokyo, 7-3-1 Hongo, Bunkyo-ku, Tokyo 113-8656, Japan
e-mail: asano@jsk.imi.i.u-tokyo.ac.jp

S. Ookubo
e-mail: ookubo@jsk.imi.i.u-tokyo.ac.jp

T. Kozuki
e-mail: kozuki@jsk.imi.i.u-tokyo.ac.jp

T. Shirai
e-mail: t_shirai@jsk.imi.i.u-tokyo.ac.jp

K. Kimura
e-mail: k-kimura@jsk.imi.i.u-tokyo.ac.jp

S. Nozawa
e-mail: nozawa@jsk.imi.i.u-tokyo.ac.jp

Y. Kakiuchi
e-mail: youhei@jsk.imi.i.u-tokyo.ac.jp

K. Okada
e-mail: k-okada@jsk.imi.i.u-tokyo.ac.jp

M. Inaba
e-mail: inaba@jsk.imi.i.u-tokyo.ac.jp

A. Bicchi and W. Burgard (eds.), *Robotics Research*, Springer Proceedings in Advanced Robotics 2, DOI 10.1007/978-3-319-51532-8_21

is often used as an indicator to prevent humanoids from falling. In general, a normal ZMP-based stabilizer often depends on 6DOF force sensors installed on both feet of the robot, which are used for computing ZMP. A malfunction of these 6DOF force sensors is not rare as they are subjected to impact loadings caused by humanoid leg movements. The breakdown of such a non-redundant sensor would be fatal because the humanoid would then fall into an uncontrollable situation. However, it is not realistic to install redundant sensors considering the sensor size and the limited design space. Moreover, the sensors restrict humanoid design flexibility as high-rated 6DOF force sensors tend to be large and this leads to a size constraint in the design process. Therefore, balancing control depending on 6DOF force sensors is an issue that should be discussed in humanoid research as a redundant installation of these sensors is difficult and the sensors malfunction.

On the contrary, musculoskeletal humanoids with human-inspired structures can adopt a new balancing strategy by utilizing their musculoskeletal structures, such as redundant sensor systems imitating the human muscle, tactile and proprioceptive receptors, or skeletal structure represented by the spine and the spherical joints. Further, balancing studies on musculoskeletal humanoids utilizing their musculoskeletal bodies have been conducted. In these studies, the researchers have attempted to achieve balancing by using the muscle reflex along with ankle and hip joints, which are joints used in human balancing strategy [7, 10].

We believe that we can build a stabilization indicator corresponding to the normal ZMP by utilizing a redundant force sensor system composed of muscle tension sensors. In other words, we can use joint torques calculated from muscle tensions instead of torques measured by 6DOF force sensors to obtain a ZMP-corresponding stabilization indicator independent of the 6DOF force sensors. Therefore, in this paper, we propose "muscle ZMP" as a stabilization indicator instead of the normal ZMP for musculoskeletal humanoids and implement the stabilizer utilizing the spine on the basis of this muscle ZMP. Further, we validate the effectiveness of muscle ZMP and the spine stabilizer by demonstrating several balancing movements of the musculoskeletal humanoid Kenshiro as shown in Fig. 1.

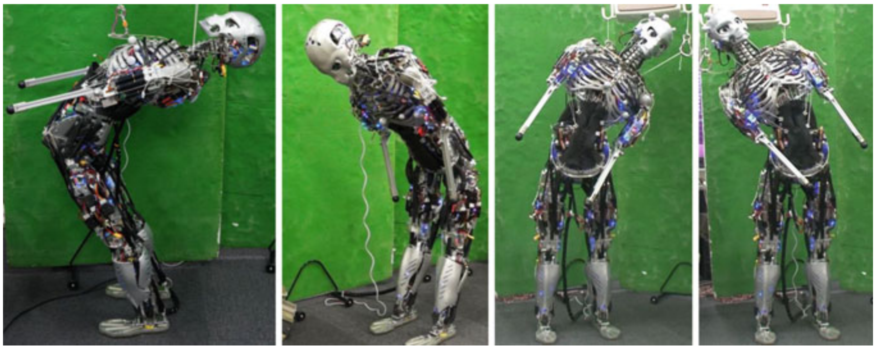


Fig. 1 Spine balancing by musculoskeletal humanoid Kenshiro

The rest of this paper is organized as follows: Sect. 2 describes the concept of the balancing strategy proposed in this paper and the musculoskeletal humanoid Kenshiro used in this study. Section 3 discusses the derivation of muscle ZMP and its verification in detail. Section 4 explains the implementation of the spine stabilizer based on muscle ZMP for musculoskeletal humanoids. Section 5 describes the balancing experiments conducted using the musculoskeletal humanoid Kenshiro to demonstrate effectiveness of the proposed balancing strategy. Section 6 presents the conclusion and briefly discusses the future research direction.

2 Balancing Strategy for Musculoskeletal Humanoids

2.1 Musculoskeletal Humanoid Kenshiro

We have developed several musculoskeletal humanoids with the purpose of building more humanlike humanoids with complex and flexible structures [5, 6, 8]. In this study, we use our latest musculoskeletal humanoid Kenshiro. Kenshiro has 64 joint DOFs in its body (excluding the DOFs of hands and the face); in particular, the spine joint is one of the unique characteristics of Kenshiro. Each joint is actuated by redundant muscle actuators. The total number of muscle actuators in Kenshiro is 105 (50 for the legs and 55 for the upper body). In terms of muscle arrangement, we selected human primal muscles that are important for fundamental motions. These muscles have the same arrangement as in humans. The left image in Fig. 2 shows the muscle arrangement in the upper body of Kenshiro, and the right one illustrates Kenshiro's spine structure. The musculoskeletal structure of Kenshiro's legs is shown in Fig. 3. Tables 1 and 2 present a list of names of the muscles in Kenshiro's upper body and legs, respectively. The joint movement range is given in Table 3.

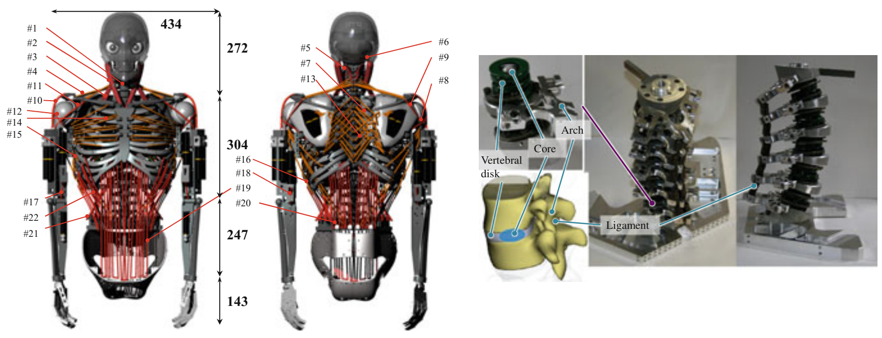


Fig. 2 *Left* Upper body of Kenshiro including the muscles [3]. *Right* Spine structure of Kenshiro. The spine is composed of five vertebrae with a combination of springs and aluminum parts [6]

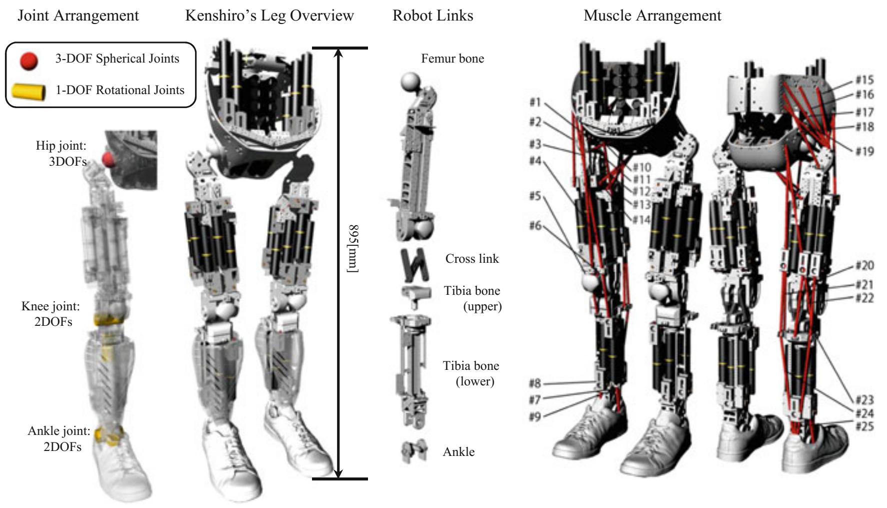


Fig. 3 Joint configuration and muscle arrangement of Kenshiro's legs

Table 1 Names of muscles in Kenshiro's upper body

#	Muscle Name	Joint
1	Longus colli	Neck
2	Sternocleidomastoid	Neck
3	Scalenus	Neck
4	Trapezius(upper)	Neck, blade
5	Splenius capitis	Neck
6	Obliquus capitis superior	Neck
7	Rhomboid	Blade
8	Infraspinatus	Shoulder
9	Posterior deltoid	Shoulder
10	Medial deltoid	Shoulder
11	Anterior deltoid	Shoulder
12	Subscapularis	Shoulder
13	Trapezius(bottom)	Blade
14	Pectoralis major	Shoulder
15	Serratus anterior	Blade
16	Latissimus dorsi	Spine, shoulder
17	Brachialis	Elbow
18	Triceps brachii	Elbow
19	Rectus abdominis	Spine
20	Erector spinae	Spine
21	Internal oblique	Spine
22	External oblique	Spine

Table 2 Names of muscles in Kenshiro's legs

#	Muscle Name	Joint
1	Iliopsoas	Hip
2	Tensor fasciae latae	Hip
3	Rectus femoris	Hip, knee
4	Sartorius	Hip, knee
5	Vastus lateralis	Knee
6	Vastus medialis	Knee
7	Tibialis posterior	Ankle
8	Peroneus longus	Ankle
9	Tibialis anterior	Ankle
10	Pectineus	Hip
11	Adductor longus	Hip
12	Adductor brevis	Hip
13	Adductor magnus (lateral)	Hip
14	Adductor magnus (medial)	Hip
15	Gluteus medius (front)	Hip
16	Gluteus medius (back)	Hip
17	Gluteus maximus (upper)	Hip
18	Gluteus maximus (lower)	Hip
19	Piriformis	Hip
20	Biceps femoris short	Knee
21	Semimembranosus/Semitendinosus	Hip, knee
22	Biceps femoris longus	Hip, knee
23	Gastrocnemius (lateral)	Knee, ankle
24	Gastrocnemius (medial)	Knee, ankle
25	Soleus	Ankle

Table 3 Joint movement range of Kenshiro

Joint*[1]		Movement range (deg)
Spine	R	−63–63
	P	−57–90
	Y	−10–10
Hip	R	−55–70
	P	−85–35
	Y	−60–60
Knee	P	−4–160
	Y	−30–40*[2]
Ankle	R	−25–12
	P	−20–60

*[1] R = roll, P = pitch, Y = yaw
*[2] Knee pitch angle = 100 (deg)

2.2 *Problem of 6DOF Force Sensors in Musculoskeletal Humanoid*

Only few 6DOF force sensors have both a high rating and a compact size that satisfy the installation requirements of Kenshiro designed to have humanlike proportions. We adopted a compact 6DOF force sensor (FTSens produced by Istituto Italiano di Tecnologia) with a diameter of 45 mm that fit the design of Kenshiro. However, this sensor has relatively low moment ratings for whole-body humanoids, and it is difficult for such compact sensors to measure high moment loads. Fig. 4 shows an example of a moment load on an ankle joint exerted when Kenshiro is standing on both legs. The sensor stops for the duration of the application of a high moment load (60–90 s). One of the reasons that the sensor stops is believed to be the hypothesis that the combination of the vertical force and the moment load is greater than the rated value during body inclination. In contrast, joint torques calculated by muscle tensions are observed normally during this period. The sensing mechanism for muscle tension does not have a problem with such loading because it has a measurable rate of about 100 kgf.

Therefore, in a musculoskeletal humanoid that requires compact 6DOF force sensors, sometimes, torques calculated by muscle tensions are more effective than those calculated by 6DOF force sensors.

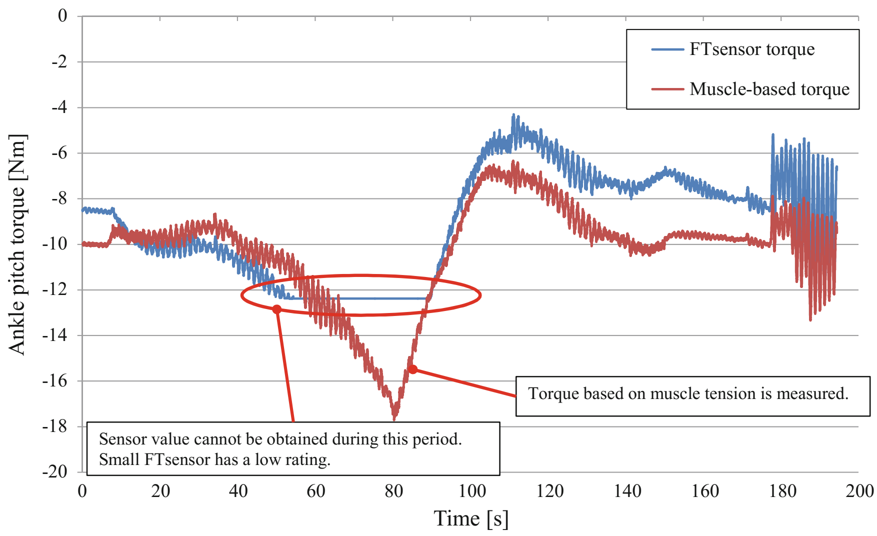


Fig. 4 Ankle joint torque measured by 6DOF FTsensor and muscle. The sensor torque cannot be measured during a high-load period around 60–90 s

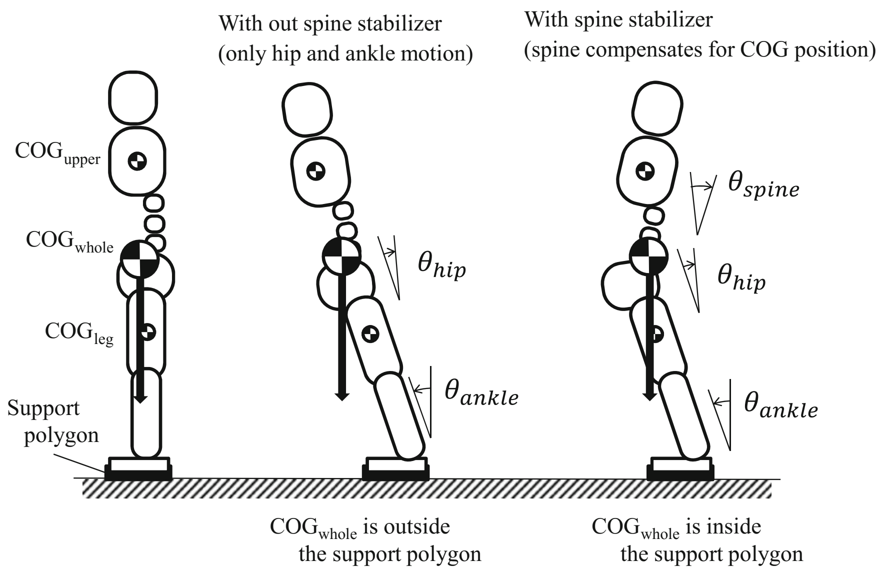

With out spine stabilizer
(only hip and ankle motion)
With spine stabilizer
(spine compensates for COG position)
$\mathrm{COG}_{\mathrm{upper}}$
$\mathrm{COG}_{\mathrm{whole}}$
$\mathrm{COG}_{\mathrm{leg}}$
Support polygon
θ_{spine}
θ_{hip}
θ_{ankle}
θ_{hip}
θ_{ankle}
$\mathrm{COG}_{\mathrm{whole}}$ is outside the support polygon
$\mathrm{COG}_{\mathrm{whole}}$ is inside the support polygon

Fig. 5 Spine stabilizer approach proposed in this paper

2.3 Spine Utilization for Balancing Stabilizer

A humanoid does not fall down when its projected center of gravity (COG) is inside a support polygon composed of foot positions and lengths. A stabilizer has to control the entire body to keep the projected COG inside the support polygon. In this study, as illustrated in Fig. 5, we compensate for the whole-body COG movement derived from the leg motion by controlling the spine to move the upper body COG so that the projected whole-body COG is inside the support polygon. It is possible to compensate for large COG movements of the leg by utilizing a wide spine movement range.

3 Muscle ZMP for Musculoskeletal Humanoid

3.1 Muscle ZMP Principle

3.1.1 Joint Torque in Musculoskeletal Humanoid

We describe joint torques on the basis of muscle tensions in a musculoskeletal humanoid. In the case of a musculoskeletal humanoid with n muscles in its ankle joint, a joint torque vector $\boldsymbol{T} (\in R^2)$ around the x and y axes is computed as follows with the condition of the muscle tension vector in the ankle $\boldsymbol{F}(\in R^n)$ and the muscle Jacobian $\boldsymbol{G}(\in R^{n\times 2})$.

$$\boldsymbol{T} = \boldsymbol{G}^T \boldsymbol{F} \tag{1}$$

$$\begin{bmatrix} \tau_{m,x} \\ \tau_{m,y} \end{bmatrix} = \boldsymbol{G}^T \begin{bmatrix} f_1 \\ f_2 \\ \vdots \\ f_n \end{bmatrix}$$

where a muscle Jacobian denotes a Jacobian matrix associating the muscle length vector $\boldsymbol{L}$ with the joint angle vector $\boldsymbol{\theta}$. The muscle Jacobian is obtained from the following equation: $\boldsymbol{G}(\boldsymbol{\theta}) = \partial \boldsymbol{L}/\partial \boldsymbol{\theta}$.

3.1.2 Computation of Muscle ZMP

In general humanoids, ZMP is computed using 6DOF force sensors installed on the robots' foot links. ZMP derivation is clearly described in [1]. Moment around the point p can be described as shown below. Its geometrical condition is illustrated in Fig. 6.

$$\boldsymbol{\tau}(\boldsymbol{p}) = \sum_{j=1}^{N} \{(\boldsymbol{p}_j - \boldsymbol{p}) \times \boldsymbol{f}_j + \boldsymbol{\tau}_j\} \tag{2}$$

where $\boldsymbol{p}$ represents the ZMP position and $\boldsymbol{p_j}$ indicates the position of force sensor. The ZMP position is calculated by solving the above equation for p_x and p_y under the condition that the right-side x and y components are zero; this condition represents a moment equilibrium around the point p. The ZMP position p_x, p_y can be described as follows:

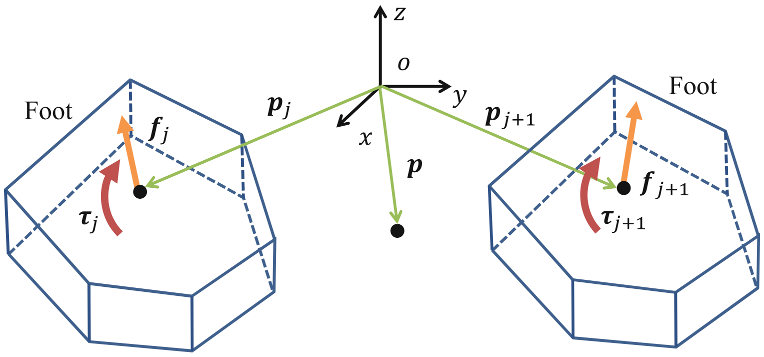


Fig. 6 Geometrical configuration for computing ZMP

$$p_x = \frac{\sum_{j=1}^{N}\{-\tau_{jy} - (p_{jz} - p_z)f_{jx} + p_{jx}f_{jz}\}}{\sum_{j=1}^{N} f_{jz}} \tag{3}$$

$$p_y = \frac{\sum_{j=1}^{N}\{\tau_{jx} - (p_{jz} - p_z)f_{jy} + p_{jy}f_{jz}\}}{\sum_{j=1}^{N} f_{jz}} \tag{4}$$

In order to obtain the muscle ZMP, we eliminate the horizontal forces at the force sensors, which are difficult to measure by using muscle tensions. In other words, we use the assumption that $f_x \approx 0$ and $f_y \approx 0$. Further, we assume that when the force sensor is close to the ground, these influences decrease. As an additional condition, we can use the ground reaction force f_z as it is needed to compute ZMP and can be measured easily by using simple an axis or three axes force sensor. Moreover, we use joint torques obtained from muscle tensions (τ_m) instead of torques obtained by the 6DOF force sensor (τ). Using these conditions, we obtain the muscle ZMP position $p_{m,x}$, $p_{m,y}$ as follows by rewriting Eqs. (3) and (4). The condition for the ankle is illustrated in Fig. 7.

$$p_{m,x} = \frac{\sum_{j=1}^{N}\{-\tau_{m,jy} + p_{jx}f_{jz}\}}{\sum_{j=1}^{N} f_{jz}} \tag{5}$$

$$p_{m,y} = \frac{\sum_{j=1}^{N}\{\tau_{m,jx} + p_{jy}f_{jz}\}}{\sum_{j=1}^{N} f_{jz}} \tag{6}$$

We consider the case of $N = 2$ for both leg force sensors, and the measurement center of f_z is located on the force sensors. The reference frame is at the middle point of the sensors. Thus, we can obtain the muscle ZMP position as follows:

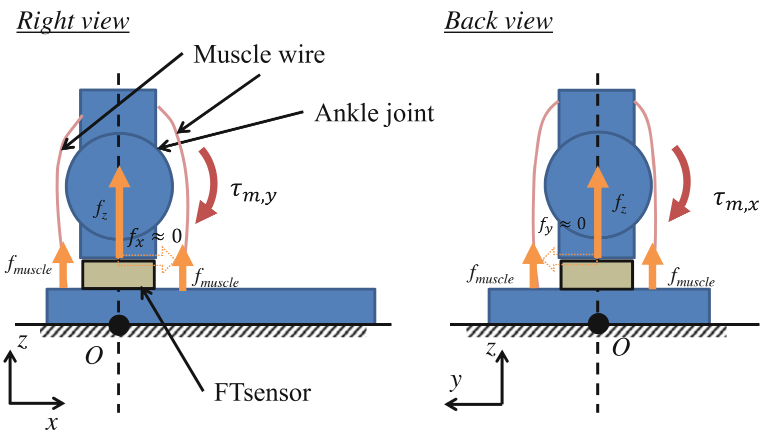


Fig. 7 Ankle geometric condition to derive muscle ZMP

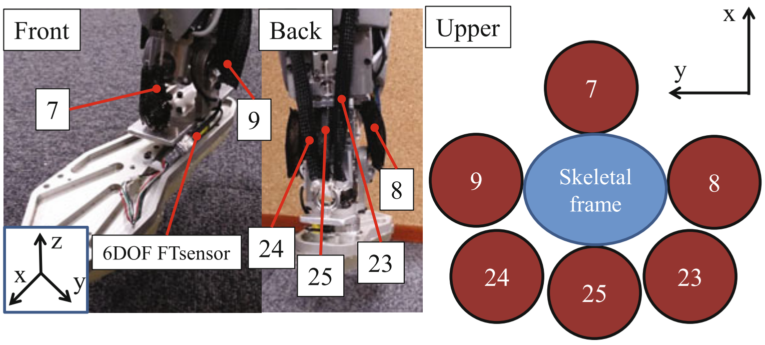


Fig. 8 Ankle muscle arrangement and 6DOF FTsensor configuration of Kenshiro. Muscle numbers correspond to those used in Fig. 3 and Table 2

$$p_{m,x} = \frac{-\tau_{m,Ly} + p_{Lx} f_{Lz} - \tau_{m,Ry} + p_{Rx} f_{Rz}}{f_{Lz} + f_{Rz}} \tag{7}$$

$$p_{m,y} = \frac{\tau_{m,Lx} + p_{Ly} f_{Lz} + \tau_{m,Rx} + p_{Ry} f_{Rz}}{f_{Lz} + f_{Rz}} \tag{8}$$

$$\boldsymbol{p}_m = [p_{m,x}, p_{m,y}, p_{m,z}]^T \tag{9}$$

In this paper, we define $\boldsymbol{p}_m$ as muscle ZMP, which is obtained using joint torques based on muscle tensions and only the ground reaction force. We use it as an indicator of stabilization control.

Kenshiro's feet have six muscles in the ankle joint and a 6DOF force sensor, as shown in Fig. 8. The ankle joint torques are obtained from these six muscles, and the 6DOF force sensor is used for obtaining only the ground reaction force f_z.

3.2 *Validity of Muscle ZMP*

We validate the characteristics of muscle ZMP by comparing it with a normal ZMP computed using 6DOF force sensors. First, we compare joint torques based on muscle tensions, with the torques obtained from the 6DOF force sensors. In an experiment, we seize one of the Kenshiro's feet in the air with tensioned muscles and load it for various directions. The result is shown in Fig. 9. Both torque values are almost the same for the entire period. Therefore, we can use the joint torques obtained from muscles instead of the torques obtained using the force sensors.

Next, we conduct an experiment to the validity of muscle ZMP against that of the normal ZMP obtained from the force sensor. In the experiment, we compare both ZMP values during Kenshiro's movement in the standing position, which is generated as its COG trajectory move along a circle. The left subfigure of Fig. 10

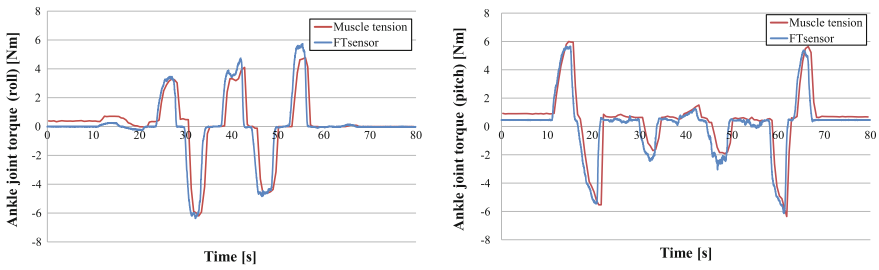


Fig. 9 Comparison of ankle joint torques obtained using muscle tensions and those obtained using FTsensor

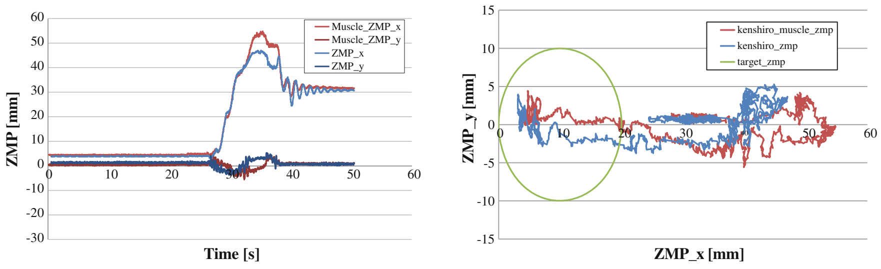


Fig. 10 *Left* Comparison of ZMP values obtained using muscle tensions and those obtained using FTsensor. *Right* The plot in the xy plane

shows the result, and thus, we can confirm that both ZMP values are almost the same. The right subfigure of Fig. 10 shows the plot of the result in the xy plane. It should be noted that low position accuracy against the target trajectory is caused by entire body flexibility of musculoskeletal humanoid.

3.3 Horizontal Force Influence to Muscle ZMP

In fact, normal ZMP includes moment components of horizontal forces f_x and f_y. In contrast, muscle ZMP does not take into account horizontal forces as in the current Kenshiro configuration including the muscle arrangement or the sensor system without the 6DOF force sensor, it is difficult to measure horizontal forces. We discuss the validity of the elimination of the moment components in this system.

We compare the torques obtained from the 6DOF force sensor and moment components calculated using horizontal forces obtained from the same force sensor. Figure 11 shows a plot of the comparison for the experiment described in Sect. 3.2. With respect to the y axis, we can eliminate the moment component generated from the horizontal force from the ZMP calculation as the y-axis torque is dominant. A percentage of the absolute min-max value between Ty_from_sensor and $Moment\_from\_fx$ is 0.0833. With respect to the x-axis, the moment component

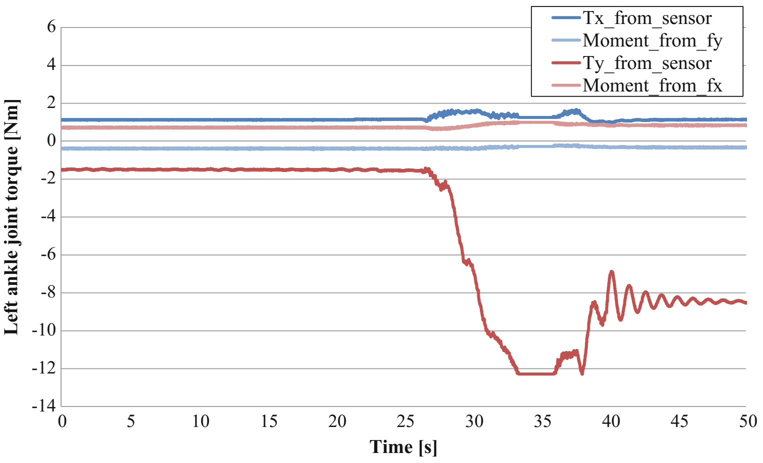


Fig. 11 Moment influence generated from f_x and f_y against FTsensor torques

is relatively large against the x-axis torque. The percentage of the absolute min-max value between Tx_from_sensor and $Moment\_from\_fy$ is 0.265. Therefore, as a characteristic of the muscle ZMP, the x position can be calculated with high accuracy. There is a possibility of low accuracy of its y position calculated using the x-axis torques. However, our motivation is to achieve humanoid motion in spite of a relatively low ZMP accuracy or not precise joint position control derived from flexible and redundant humanlike structures. In order to obtain more accurate muscle ZMP, one of the methods is to use three axes force sensors for taking horizontal forces into the calculation process of the muscle ZMP.

4 Spine Stabilizer for Balancing Control

4.1 Spine Stabilizer Overview

We now present an overview of the spine stabilizer system implemented in this study in Fig. 12. At the beginning of the stabilizer sequence, the humanoid is set in a standing position on the ground and we obtain muscle ZMP as the reference ZMP. There is a component to estimate joint torques in the upper part of the control cycle. This component estimates the joint torques of each joint on the basis of relative muscle variations and muscle tensions. In the layer of the spine stabilizer, the stabilizer computes the muscle ZMP and the target spine angle. The humanoid moves according to the decided spine angle, and then, the muscle lengths, muscle tensions, and ground reaction forces are updated. Further, the spine stabilizer can compensate for humanoid motions generated from joint angle commands sent from the upper layer.

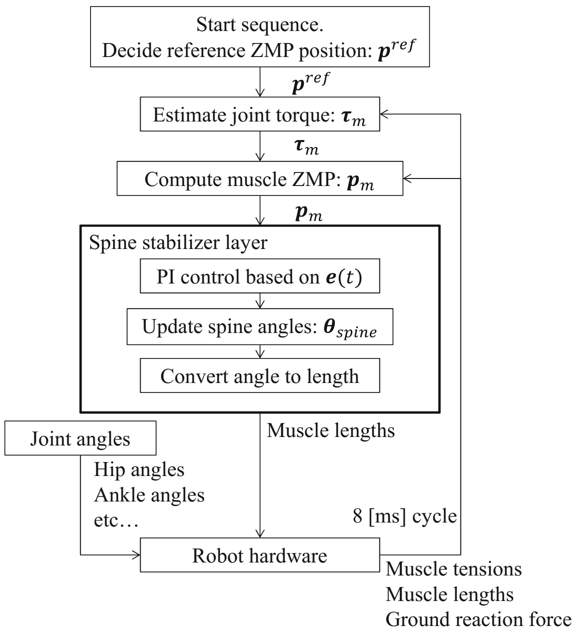


Fig. 12 Flowchart of the spine stabilizer sequence

4.2 Implementation of Spine Stabilizer

The spine stabilizer is implemented to compensate for humanoid imbalance by making spine roll or, pitch movement. The movement is generated by the PI control of the spine angle on the basis of an error between the reference ZMP and the muscle ZMP of this cycle. Muscle lengths related to the decided spine angles are computed and sent to the humanoid. Thus, we control the spine as follows:

$$\theta_{spine} = K_p e(t) + K_i \int e(t)dt \tag{10}$$

$$\theta_{spine} = \begin{bmatrix} \theta_{roll} \\ \theta_{pitch} \end{bmatrix} \tag{11}$$

$$e(t) = \begin{bmatrix} p_y^{ref} - p_y(t) \\ p_x^{ref} - p_x(t) \end{bmatrix} \tag{12}$$

where $e(t)$ denotes the ZMP error at each control cycle. Further, p_x^{ref} and p_y^{ref} represent the x and y components of reference ZMP, respectively. As the reference ZMP, we use values at the beginning of the stabilizer sequence. Then, we implement this sequence in the layer of the 8 ms control cycle.

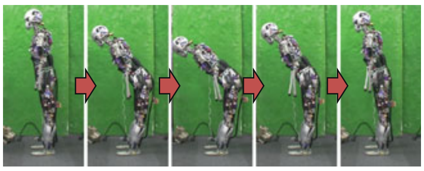
5 Balancing Experiment with Spine Stabilizer Based on Muscle ZMP

We conducted balancing experiments utilizing the two components proposed in the previous sections. The first is component is the muscle ZMP used as a balancing indicator instead of the normal ZMP. The other is the spine stabilizer based on the PI control of spine angles.

5.1 Forward and Backward Balancing

We conducted forward and backward balancing experiments in the sagittal plane. In both experiments, the ankle and hip joint angles were commanded from the upper layer and the spine stabilizer compensated for the imbalance of Kenshiro. The upper layer command was determined by the operator. During the experiment, the head and pelvis angles of Kenshiro could be measured from the IMUs installed on each link.

The left image of Fig. 13 shows the setup of the forward balancing experiment, and the right image illustrates the plot of the experimental data. We can confirm that the spine stabilizer works well in terms of the behavior of the muscle ZMP, which seemed to follow the reference ZMP. Moreover, we can confirm a large body bending movement from the maximum head pitch angle of 54.0 deg. Figure 14 shows the setup of the backward balancing experiment and a plot of the experimental data. We can also confirm the stabilizer's contribution to balancing by observing the similar behavior of the muscle ZMP that followed reference ZMP. Further, we can confirm a large body extension movement from the minimum head pitch of −54.4 deg.



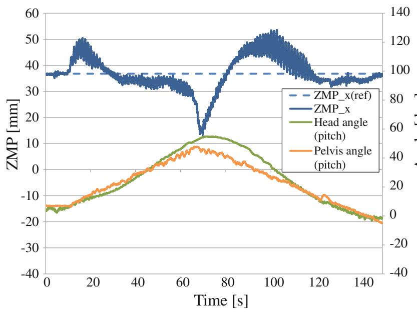
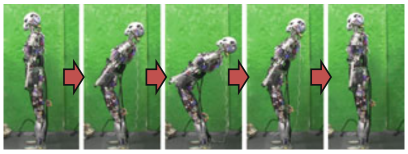


Fig. 13 *Left* Forward flexion movement by musculoskeletal humanoid Kenshiro. *Right* Experimental data during the forward flexion movement of Kenshiro



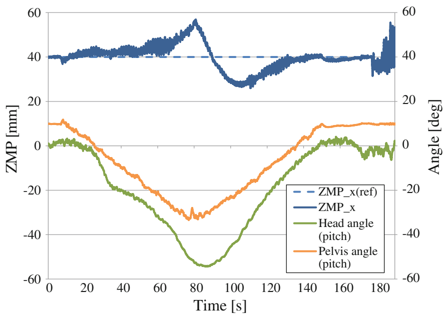
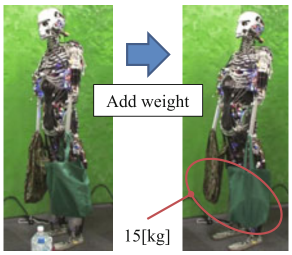


Fig. 14 *Left* Backward extension movement by musculoskeletal humanoid Kenshiro. *Right* Experimental data during the backward extension movement of Kenshiro



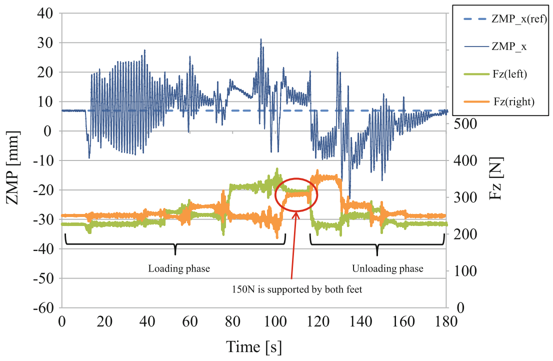


Fig. 15 *Left* Weight addition experiment with the spine stabilizer. A 15-kgf weight is added, and the result are shown in the right image. *Right* Experimental data during the weight addition experiment

5.2 Balancing with Weight

We conducted an experiment to validate the loading durability. We added weights to the bags attached to the end effectors of Kenshiro. Kenshiro could remain in a standing position with a 15 kgf weight.

The left image of Fig. 15 shows the experimental setup, and the right shows a plot of the experimental data. We can confirm that the spine stabilizer works well. In terms of the ground reaction force, we can confirm that a load of about 15 kgf (147 N) is exerted on Kenshiro because the total ground reaction force is 477 N at 0 s and 622 N at 110 s.

6 Conclusion and Future Works

In this paper, we proposed a new balancing strategy for musculoskeletal humanoids with redundant force sensors. The purpose of this study was the exploration of a new humanoid control strategy compared to an ordinary strategy based on an accurate ZMP measurement and precise position control.

We proposed the use of muscle ZMP as a stabilization indicator instead of the normal ZMP based on 6DOF force sensors by utilizing muscle tensions acquired from the robot's muscle actuators. From the validation experiment of muscle ZMP, we confirmed that torques obtained from muscle tensions are almost the same as those obtained from 6DOF force sensors. Moreover, we validated the muscle ZMP value by comparing it with a normal ZMP. Furthermore, we implemented a muscle ZMP-based spine stabilizer for compensating humanoid imbalance. By using the stabilizer, we demonstrated the forward and backward bending motions of Kenshiro and its loading durability with about 15 kgf weights. Although a musculoskeletal humanoid is seemingly complex with redundant body structures, we demonstrated that it is possible to construct a novel whole-body control strategy for humanoids by using their complexity effectively.

In the future, we would like to extend the use of muscle ZMP as a more robust indicator for stabilization by considering the friction between a robot's feet and the ground. Moreover, we would like to conduct the verification not only under static conditions but also in dynamic situations. We would also like to integrate the balancing strategy with a whole body motion generation software program on the basis of the abovementioned attempts. We believe that this will lead to the formulation of a new humanoid control strategy based on humanlike flexibility or redundant body structures that allow low-accuracy joint position control or robot motion under uncertain environmental conditions.

References

1. Kajita, S., Hirukawa, H., Harada, K., Yokoi, K.: Introduction to Humanoid Robotics. Springer, Berlin Heidelberg (2014)
2. Kajita, S., Morisawa, M., Miura, K., Nakaoka, S., Harada, K., Kaneko, K., Kanehiro, F., Yokoi, K.: Biped walking stabilization based on linear inverted pendulum tracking. In: Proceedings of the 2010 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS'10), pp. 4489–4496 (2010)
3. Kozuki, T., Motegi, Y., Shirai, T., Asano, Y., Urata, J., Nakanishi, Y., Okada, K., Inaba, M.: Design of upper limb by adhesion of muscles and bones -detail human mimetic musculoskeletal humanoid Kenshiro. In: Proceedings of the 2013 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS'13), pp. 935–940 (2013)
4. Li, Z., Vanderborght, B., Tsagarakis, N.G., Colasanto, L., Caldwell, D.G.: Stabilization for the compliant humanoid robot COMAN exploiting intrinsic and controlled compliance. In: Proceedings of The 2012 IEEE International Conference on Robotics and Automation, pp. 2000–2006 (2012)

5. Mizuuchi, I., Nakanishi, Y., Sodeyama, Y., Namiki, Y., Nishino, T., Muramatsu, N., Urata, J., Hongo, K., Yoshikai, T., Inaba, M.: An advanced musculoskeletal humanoid Kojiro. In: Proceedings of the 2007 IEEE-RAS International Conference on Humanoid Robots (2007)
6. Nakanishi, Y., Asano, Y., Kozuki, T., Mizoguchi, H., Motegi, Y., Osada, M., Shirai, T., Urata, J., Okada, K., Inaba, M.: Design concept of detail musculoskeletal humanoid "Kenshiro" toward a real human body musculoskeletal simulator. In: Proceedings of the 2012 IEEE-RAS International Conference on Humanoid Robots, pp. 1–6 (2012)
7. Nakanishi, Y., Namiki, Y., Hongo, K., Urata, J., Mizuuchi, I., Inaba, M.: Realization of large joint movement while standing by a musculoskeletal humanoid using its spine and legs coordinately. In: Proceedings of the 2008 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS'08), pp. 205–210 (2008)
8. Nakanishi, Y., Ohta, S., Shirai, T., Asano, Y., Kozuki, T., Kakehashi, Y., Mizoguchi, H., Kurotobi, T., Motegi, Y., Sasabuchi, K., Urata, J., Okada, K., Mizuuchi, I., Inaba, M.: Design approach of biologically-inspired musculoskeletal humanoids. Int. J. Adv. Robot. Syst. **10**, 1–18 (2013)
9. Nishiwaki, K., Kagami, S.: Strategies for adjusting the ZMP reference trajectory for maintaining balance in humanoid walking. In: Proceedings of The 2010 IEEE International Conference on Robotics and Automation, pp. 4230–4236 (2010)
10. Ogawa, K., Narioka, K., Hosoda, K.: Development of whole-body humanoid "Pneumat-BS" with pneumatic musculoskeletal system. In: Proceedings of the 2011 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS'11), pp. 4838–4843 (2011)
11. Ott, C., Roa, M.A., Hirzinger, G.: Posture and balance control for biped robots based on contact force optimization. In: Proceedings of the 2011 IEEE-RAS International Conference on Humanoid Robots, pp. 26–33 (2011)

How to Think About Grasping Systems - Basis Grasps and Variation Budgets

Leif P. Jentoft, Qian Wan and Robert D. Howe

1 Grasping Systems and Variation

Creating versatile grasping capabilities is a longstanding challenge in robotics. Although robots grasp effectively in structured factories, they need to be more versatile to handle objects in unstructured environments where many factors affect grasp success, including a wide range of object shapes and sizes, incomplete and frequently inaccurate perception, uncertainties in surface friction and mass, and robot positioning errors. The high-dimensionality of the problem makes it difficult to understand the capabilities and limitations of grasping systems. Analytical methods (such as grasp simulation and manipulability analysis) are limited because real environments contain too many objects described by too many parameters for tractable evaluation. Standardized object sets enable experimental comparison of the performance of different systems, but it is not straightforward to extrapolate from such experiments to predict performance on novel objects. Thus there is a lack of effective system-level metrics, and this poses a major barrier to progress because understanding the capabilities and limitations of grasping systems is essential for comparing the benefits of different approaches, and for evaluating design tradeoffs within and between robot subsystems. As a result, robotics researchers must currently direct their efforts based on intuitive analysis of prior results.

The goal of this paper is to develop a framework for understanding grasping system performance and for designing capable systems. In the first half of this paper,

L.P. Jentoft (✉)
RightHand Robotics Inc, Cambridge, MA, USA
e-mail: leif@righthandrobotics.com

Q. Wan · R.D. Howe
Harvard School of Engineering and Applied Sciences, Cambridge, MA, USA
e-mail: qwan@seas.harvard.edu

R.D. Howe
e-mail: howe@seas.harvard.edu

A. Bicchi and W. Burgard (eds.), *Robotics Research*, Springer Proceedings
in Advanced Robotics 2, DOI 10.1007/978-3-319-51532-8_22

we cast the grasping problem as *overcoming variation* and project it onto a traditional robot subsystem decomposition. This forces explicit examination of which sources of variation matter, and provides a way to understand the tradeoffs between alternate ways to address the variation, which is particularly useful to compare the performance of disparate systems.

In the second half of this paper, we use this approach to build a methodology for designing grasping capabilities. First, we start with a *basis grasp*: a specific finger configuration on a specific object. Second, we design a combination of motion sequences, sensing, and passive mechanics to make grasp acquisition robust to variations in object shape and pose, perception, and robot control. Third, we analyze the basis grasp's robustness to local variation to determine the *basin of attraction*, the range of variation it can tolerate while still achieving a good grasp. Finally, we treat this basin of attraction as a *variation budget* that can be distributed across subsystems to inform system tradeoffs between perception errors, robot inaccuracies, and object variation. To extend system capabilities to a greater range of objects and variations, additional basis grasps can be added. The principle advantage of this approach is that within such a specific context, the effects of local variations can be understood, as well as quantified and therefore compared across disparate systems.

2 Posing the Grasping Problem as Overcoming Variation

The ultimate goal is to build grasping systems that work everywhere, on everything. The challenge is overcoming variation, which comes from a wide range of sources, including object diversity in shape, friction, mass, and pose; perceptual variability due to limited camera resolution, segmentation errors, and occlusion; robot arm and finger positioning errors; noise and sensitivity limits in force sensors; and many others. In this section, we present an overview of how the subsystems of a robot grasping system work together to deal with variability. This provides a consistent way to understand the relative advantages of different approaches and to understand the tradeoffs within subsystems, enabling incremental progress in the development of grasping systems.

2.1 System Breakdown

As a foundation for analysis, it is helpful to break out the typical subsystems of a robotic grasping system as described in Fig. 1. This, of course, shows only the major interactions (real systems have more complex information flow), roughly following the classical "sense - think - act" structure.

The *Task Interface* presents the robot's general capabilities to a user so they can engage it to perform a specific task. This can be very simple – how to move individual robot joints – or more complicated – what objects are perceived by the robot, how

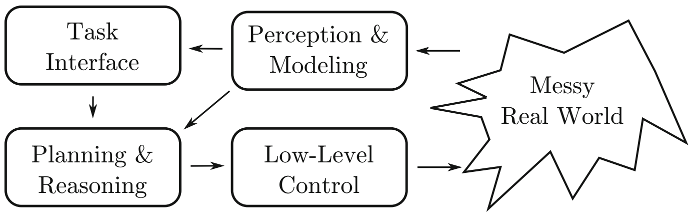


Fig. 1 A typical system breakdown for a grasping robot. The task interface is used to direct the robot's general capabilities to a specific task, setting the required parameters. The perception and modeling system takes raw sensor data from the real world and uses it to synthesize an internal model. The planning and reasoning system uses this model to map the task parameters to the sequence of commands executed by the low-level control, and (if necessary) change the plan based on new feedback from the perception/modeling system

to grasp them, etc. Robots do not need to autonomously compensate for all sources of variation to be useful, but the more they can overcome automatically, the simpler the task interface is and the better they can function outside static environments.

The *Perception System* gathers and interprets data from the messy real world to create an internal model of the object to be grasped and the surrounding environment. This can both remove variation by creating an accurate internal model, and introduce variation through perceptual inaccuracies. The more detailed the model, however, the more difficult or time-consuming it is to create: a simple 2D view of the facing side of an object is easier to obtain than a precise 3D geometric model that includes the object's far side.

The *Planning-Reasoning System* plans low-level actions such as where to place fingers on an object to overcome variation in shape or pose, and how to sequence corrective actions. It bases these plans on the model created by the perception system, information from the task interface, and any *a priori* knowledge.

The *Low-Level Control* system is the interface to interactions with the external world, such as arm and hand hardware and closed-loop controllers for joints, and passive or compliant mechanisms that automatically adapt to limited ranges of external variations. Choosing the appropriate basis for this control has a large impact on the level of variation tolerated from the rest of the system – stiff position-controlled actuators exert large forces in response to positioning errors from the perception system, whereas force-control loops may require more nuanced reasoning about how to use environmental affordances to maintain stability.

2.2 Robot Grasping Results Viewed in Terms of Variation

Using this framework, prior research in grasping, albeit on diverse and seemingly unrelated topics, can all be seen as working towards coping with variation.

Traditional industrial applications of robots use careful structuring of the environment and heavy, stiff robots to eliminate variation in the object and in robot motion. This severe restriction on object and environment variation allows industrial application to use simple perception, planning, and control systems. Any variation from one object to another, such as switching the production line to a new product, must be addressed through the task interface. Typically, this requires a highly-trained technician to use low-level programing or a teach pendant to reconfigure the system for each new object.

Simulation-based planners such as GraspIt [22] and OpenRave [5] compensate for variations in object geometry and pose by finding the right locations to place fingers to achieve a good grasp. Many different hand poses are sampled, and their quality is evaluated using grasp metrics such as as *epsilon quality* [10] and reachability. These planning systems place a large burden on the perception system because they require a precise, complete model of the object geometry, so, for example, the perception system must fill in raw sensor data by fitting object models from *a priori* object libraries to clusters of points. Most simulation-based planning approaches do not compensate for variations due to inaccuracies in the perception or robot control systems, though recent work by Weitz et al. [29] incorporates this into the grasp quality metric.

Grasp site strategies compensate for variations in object pose and geometry by searching for consistent grasp sites on varied objects. This simplifies the perception system because it removes the need for detailed or *a priori* object models. Instead, this approach attempts to find acceptable grasp sites directly in raw perception data. Saxena et al. search for grasp sites directly in 2D image data [27]. By manually labeling the grasp points for a parallel gripper on a set of objects in simulation, they create visual classifiers for grasp sites by simulating scenes under a wide range of poses and lighting conditions. These classifiers perform well on novel objects outside of simulation. Working with laser range data, Klingbeil et al. use a template to search for regions that match the shape of a parallel-jaw gripper [17]. Herzog et al. present a more generalized approach in a similar vein [12] based on a general grasp site template searched across orientations. This allows the re-use of more complicated grasps from human demonstrations, and results are presented using both a parallel-jaw gripper and a Barrett Hand in two different preshapes. The existing literature does not show how much variation is tolerated in the identified grasp sites, but the overall performance of such systems is strong.

Heuristic grasp planners use empirical rules to determine where to place a hand to compensate for varied geometry and pose. For example, Hsiao et al. create a set of candidate grasps around stereotyped poses and score them based on factors such as the quality of perception data at the grasp site, their likelihood to cause the object to be knocked over, and their proximity to the current position of the gripper [13]. This approach also reduces demands on the perception system, as detailed object models are not required. Understanding the capabilities and limitations of these systems is challenging because it is difficult to connect the collection of heuristics to the range of variation in object shape and pose where they are successful; most papers only characterize system performance against *ad hoc* collections of objects.

Anthropomorphic hands are perhaps the most complex examples of the low-level control system in Fig. 1. These hands attempt to mimic human functionality with three to five highly-dexterous fingers that can exert contact forces in any direction [4, 20, 23]. In principle, the many degrees of freedom in these hands can be used to cope with a wide range of object variation. Unfortunately, understanding how to use this complexity in unstructured grasping has proved elusive. A number of factors contribute to the challenges. The needed interactions with planning and perception systems have not been successfully defined or implemented. There is a considerable body of theoretical work that seeks to compensate for variations in object geometry and task constraints by controlling contact forces; a good review is presented by Shimoga [28]. However, although this provides an elegant way to understand the role of geometric variation, low-level control of these complex machines has been limited by factors such as friction, tendon dynamics, and poor contact sensing. Anthropomorphic hands have rarely been used outside of controlled research settings.

Underactuated hands compensate for variations in object pose, object geometry, perception errors, and arm positioning errors by mechanical design [1, 2, 8, 18]. Compliance in the fingers allows them to passively adapt to the details of the object geometry, and thereby reduces the load on both the perception and planning systems. [7]. Recent work such as the coin-flip primitive presented by Odhner et al. in [19] has extended this approach beyond grasping into manipulation.

The final examples examined here come from three teams in the DARPA Autonomous Robot Manipulation competition that developed systems to perform a set of pre-specified tasks with a known set of objects and tools [11]. These are among the best-integrated and autonomous grasping systems presented to date, so their approach to dealing with variability is of particular interest.

The system created by Hudson et al. [14] primarily used the perception system to overcoming variations in robot arm positioning and camera registration. They modeled the difference between the arm's actual pose and expected pose using an unscented Kalman filter, and made extensive use of *a priori* object models to compensate for occluded camera views. This effectively compensated for variations from both the low-level control system (which introduced positioning errors up to several cm) and from the perception system, and the team achieved top scores in the competition. It provided only a limited solution to object variation; the grasp planner used a full 3D model of each object to create a library of grasp candidates by simulating which hand placements maximize contact surface, and the resulting grasp candidates were manually pruned for each object.

The system created by Schaal et al. [26] primarily used the low-level control system to overcome variation in the arm positioning and object geometry and pose. In their approach, grasping is reformulated from the position domain to the force domain using "Dynamic Motion Primitives" (DMPs). Because the DMP only requires a few parameters, this formulation also enables the effective use of machine learning to optimize the grasping plans. The plans themselves are created from demonstration. Because force-domain execution requires less information about the object than position-domain execution, this approach is more readily adapted to unknown objects. Although *a priori* object models are used in [26] in a manner similar to

Hudson et al.'s approach (using iterative-closest-point matching to align model and sensor data), the team was subsequently able to extend it to a model-free approach [12]. An extensive calibration routine is required to compensate for variations in the response of the strain gauges used to measure force.

Bagnell et al. [3] overcame variation by detecting errors and sequencing corrections using behavior trees implemented in a framework called the "Behavior Architecture for Robotic Tasks" (BART). This approach relied on creating a good task interface to sequence and combine primitives in the planning-reasoning system.

Thus these three teams focused on different subsystems in their solutions, with the first focusing on the perception system, the second on the low-level control subsystem, and the third on the task interface and planning-reasoning subsystems. By considering the mechanisms for coping with variability, we can understand why these teams achieved roughly comparable performance despite the use of radically different approaches.

3 Basis Grasps and Variation Budgets

We can also apply the framework prospectively to design and analyze new robot grasping capabilities, again defining grasping capability in terms of the ability to successfully execute a grasp across variation (in object geometry, perceptual noise, etc.). Under this definition, the key challenge to creating broader functionality is to understand what variation matters for achieving a successful grasp, and to design systems that compensate for it. To do so, we invert the usual order: rather than starting with an object and determining how to grasp it, we start with a *basis grasp*, a specific finger configuration, and determine the range of object variation where it will work. Second, we enlist the entire robot (perception, planning, low-level control systems) to make this grasp tolerate local variation and still achieve a successful grasp. Third, we analyze the bounds of this variation to determine the *basin of attraction* around the template configuration. This is both a measure of grasping capability, and a metric for where the grasp can be successfully applied. To extend the range of object variation that can be grasped, we can create a collection of basis grasps with different basins of attraction.

The principle advantage is that variation is easier to understand when examined locally as deviation from a basis grasp. This means it is faster to establish which sources of variation are dominant in determining a grasp's success. It is also easier to see how to cope with variations using a robot's full capabilities, and it is more tractable to establish bounds for the system's ability to grasp related objects. In the following section, several examples are presented to illustrate the framework.

3.1 Overhead Three-Fingertip Grasp

In the first example, we study the i-HY hand [25] (Fig. 2) in an overhead fingertip grasp on a box-shaped object sitting on a table (Fig. 3). This hand has three compliant, underactuated fingers, each controlled by a separate actuator, along with a fourth actuator that controls the orientation of the two fingers. Tactile sensors are located on the fingers, and the proximal joints are equipped with magnetic encoders; the deflection of the distal joints can be determined from the excursion of the tendon measured at the proximal joints and at the spools on the actuators. In this basis grasp, the fingers are placed on antipodal surfaces of the object.

Determining the object variation range. Now, we analyze the dominant types of variation that limit successful grasps. The point of this analysis is not to demonstrate a method that overcomes any arbitrary source of variation, but to show how such analysis can be used to easily understand the capabilities and limits of a given grasping

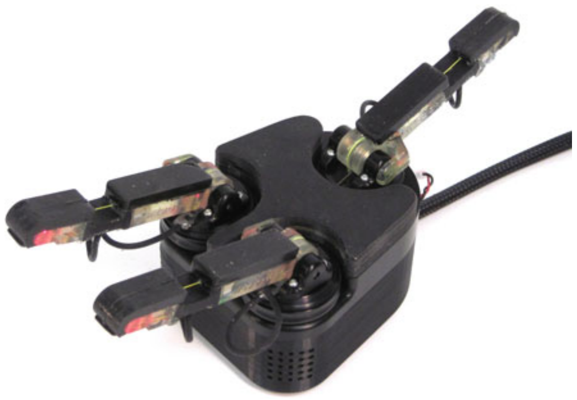


Fig. 2 The i-HY hand

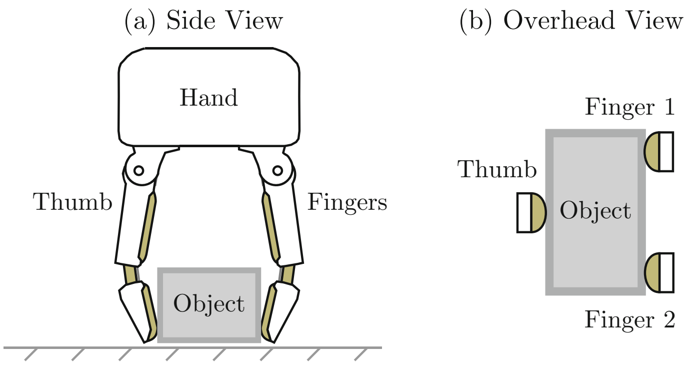


Fig. 3 An example basis grasp: the overhead fingertip grasp on a rectangular prism **a** side view **b** overhead view

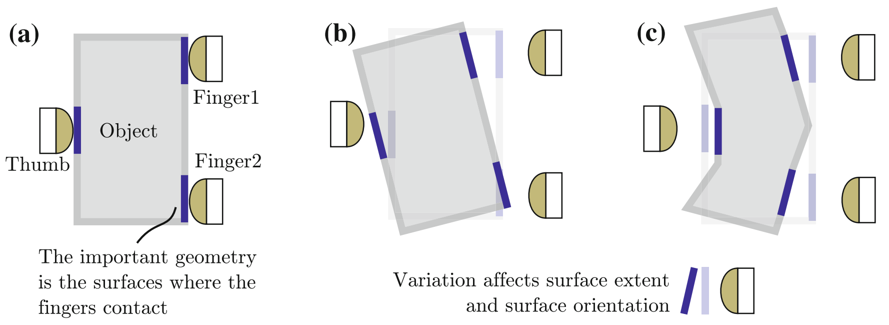

(a)
Object
Finger1
Thumb
Finger2
The important geometry is the surfaces where the fingers contact
(b)
(c)
Variation affects surface extent and surface orientation

Fig. 4 **a** The important part of an object's geometry is the place where fingers contact the object. This can be used to parameterize variations due to **b** object pose and robot registration and **c** object geometry and imperfect visual segmentation

system. In this grasp (as in many), the dominant factor is object geometry and object pose. The basis grasp is defined with the fingers well-aligned with the hand (Fig. 4a), but if the object pose is rotated due to inaccuracies in the perception or control systems, finger contact locations will be displaced and rotated (Fig. 4b). Simple analysis of finger motions and surface normals can then reveal the range of pose variation where this grasp will succeed.

Similarly, if the object shape is not a rectilinear box, the grasp may still succeed. The key observation is that the only part of the object geometry that affects grasping is the contact surface patches where the fingers make contact (Fig. 4c). Thus when the grasp is used as the reference frame (rather than the object, as in traditional grasp analysis), all geometric variations from object, robot control, and sensing can be condensed into one quantity: the local variation in the surface patches where fingers contact the object. Once again, analysis of finger motions and surface normals will specify the range of shape variation (and combination of shape and pose variation) where this basis grasp will succeed.

Extending the grasp variation range. To make this grasp more robust to local variation, we then enlist the other subsystems of the robot, particularly the low-level control system. One variation that is important to take into account is vertical position of the object, due to errors in the perception system, mis-calibration of the robot arm with respect to the vision system, robot control errors, etc. We can compensate for vertical variation by referencing the finger pose to the table supporting the object (Fig. 5). This is done by with a guarded move from above (i.e. approach-until-contact), using tactile sensors in the finger tips to determine when contact occurs. This eliminates the need for precise estimation of the height of the object from the perception system. We also slide the fingers along the table surface as they close – this approach uses the compliance of the fingers to compensate for any minor variation in vertical position that might allow thinner objects to slip underneath the fingertips as they close.

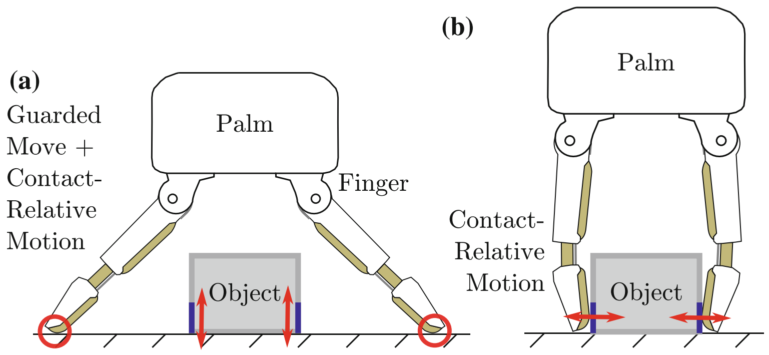


Fig. 5 Sensing, control, and targeted mechanical design can be used expand the basin of attraction. For the surface grasp, **a** a guarded move against the supporting surface is used to compensate for variation in the contact surface height, and **b** contact-relative motion around the object surface is used to compensate for variation in the contact surface extent

We can extend the basis grasp's tolerance to variation in the width of the object (i.e. the contact surface patches) by again using a guarded move. After the fingers contact the table surface, the hand is lifted incrementally while maintaining fingertip contact. When the tactile sensors in the distal link signal contact with the side of the object, the controller can shift from closing the fingers to increasing grasp force. Alternatively (or in addition), the joint position signals can indicate that the fingertips have stopped closing. Note that these strategies for dealing with variation in both vertical height and width are based in strategic use of low-level control – neither guarded moves or compliant contact require detailed information from the world model created by the perception system.

Having defined the basis grasp in terms of finger configuration as well as low-level control behavior, we can establish quantitative bounds on how much variation can be tolerated for each important parameter of variation. The fingers must contact the object as they close, which means the object width must fit inside the fingers in order for the acquisition strategy to succeed (Fig. 6-left), and the object must extend laterally past the two adjacent fingers (Fig. 6-right). This forms a performance bound on how much variation in object size the grasp can tolerate, as shown in the shaded region in Fig. 6. Similar analysis can be applied to variation in object orientation, friction, mass, etc. – where selection of factors to include is a function of the dominant balance in a given grasp. We propose the term *basin of attraction* to describe the range of variation the grasp tolerates.

A simple experiment was performed to illustrate this approach, as shown in Fig. 7. A small object (an allen key set, approximately 25 x 25 x 75 mm) was placed on a table and the hand executed the overhead fingertip basis grasp. This process was repeated as the hand was shifted in each direction. Fig. 7-left shows the results for shifting in the width direction, and Fig. 7-right shows the results for shifting laterally. In each plot, the height of the red line above the displacement axis indicates the region

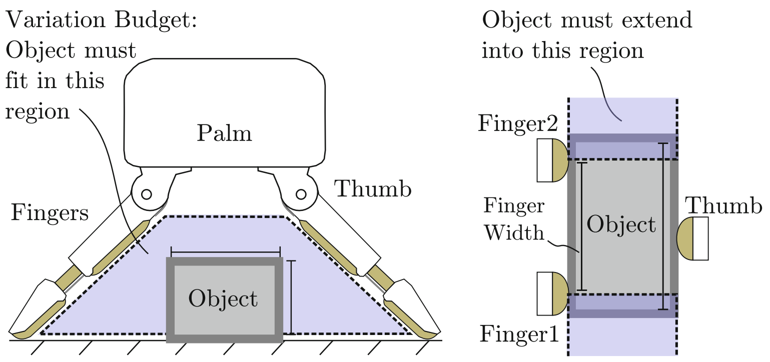

Variation Budget: Object must fit in this region
Palm
Fingers
Thumb
Object
Object must extend into this region
Finger2
Finger Width
Object
Thumb
Finger1

Fig. 6 The basin of attraction for the overhead fingertip grasp when the object is centered in the grasp

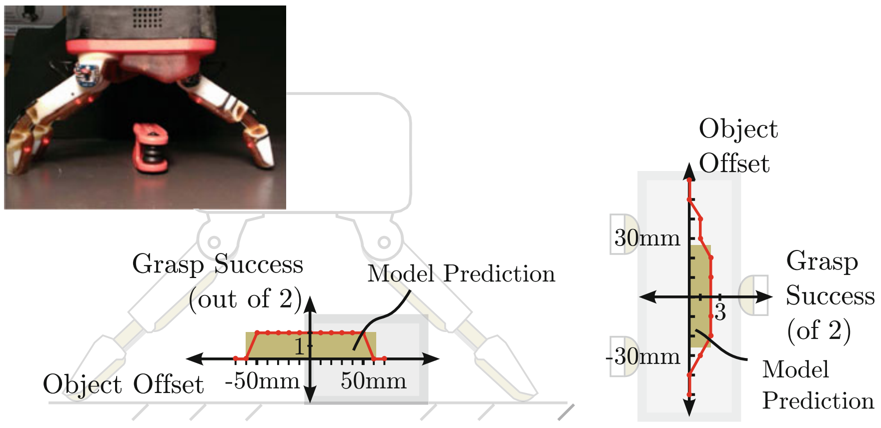


Fig. 7 Experimental validation of the basin of attraction closely matches predicted results. A small object (allen key set) was grasped under a variety of positioning offsets to determine the bounds on the basin of attraction

of grasp success, which closely corresponds to the simple analysis predicting grasp success.

Variation budgets. Now that the limits to variation have been determined, this basin of attraction can be treated as a *variation budget* that can be allocated to the diverse sources of variation for a particular application (Fig. 8). For example, the uncertainties due to limitations in the visual perception and robot control subsystems can be determined, and subtracted from the total basin of attraction. The remaining region then defines the range of object variation that the system will be able to deal with effectively - i.e., the overall system's variation performance. This approach makes it possible to evaluate quantitative tradeoffs between different subsystems and determine, for example, the impact of low-precision arm control or high-resolution

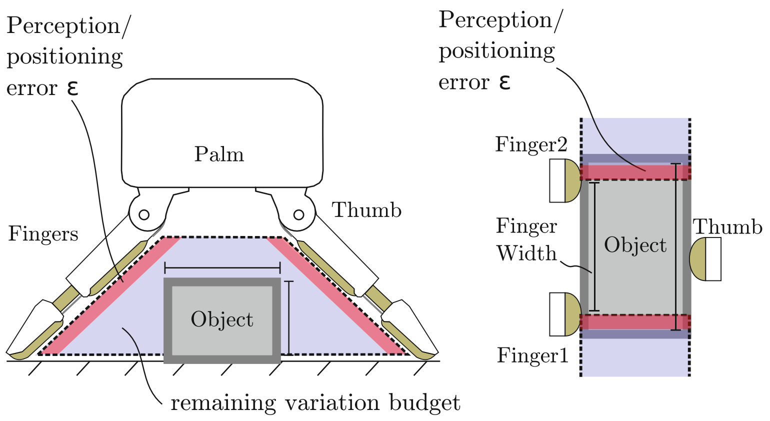


Fig. 8 The basin of attraction serves as a variation budget that can be spent on different subsystems. Here uncertainties due to perception and robot control are represented as the *red regions* that shrink the *shaded region* that is available to deal with object variations

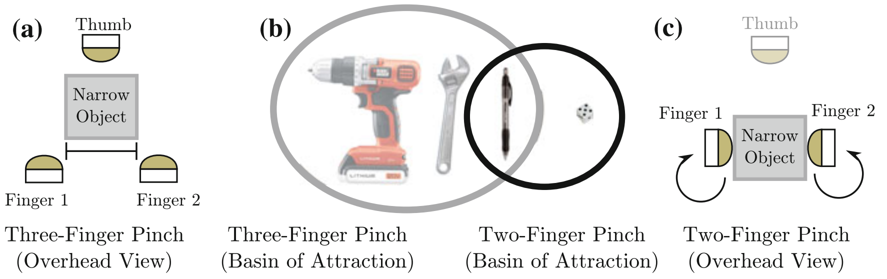


Fig. 9 Building a collection of basis grasps. **a** The Overhead Three-fingertip Grasp does not cover sufficient object variation to grasp narrow objects. **c** A robot's skills can be augmented by adding additional basis grasps, such as the two-fingered pinch. **b** The central panel shows the basin of attraction (*circles*) for each of the two grasps; the region of intersection includes objects that can be successfully grasped with either basis grasp

RGB-D imaging on the range of objects that can be grasped. It can also be used to compare different grasping strategies and grasping systems.

3.2 Other Basis Grasps

A single basis grasp spans only a limited (but defined) range of objects; a collection of them can be used to provide wider capabilities. For example, the Overhead Three-fingertip Grasp cannot grasp objects smaller than the spacing between the adjacent fingers (Fig. 9a). However, another primitive can be constructed based around the pinch configuration, with the two fingers rotated so that they meet in the center (the thumb is not used), as shown in Fig. 9c. This extends the hands capability for grasping small objects. The same approaches can be used to generate tolerance of local variation (guarded moves, compliance), but note that there is a different dominant balance for which variation is important for this grasp. Two opposing fingers are less able to resist moments caused by offset center of mass, so the object's mass and alignment with the center of mass matter more than with the three-fingered grasp.

4 Discussion

The goal of this paper is to present a way to reason about dominant effects in the messy problem of robot grasping. Despite a significant effort to find a unified theoretical framework for grasping, none has achieved widespread success. This is perhaps not surprising given the complexity of the physical phenomena involved in robotic grasping – it involves incomplete perceptual data, complex interaction mechanics (varied surface friction, compliance, closed-loop kinematic chains), varied boundary conditions (clutter, affordances), and an arbitrary range of object geometries. The key to creating effective functionality in the near term is understanding where the problem can be condensed, and how to quantify the condensed functionality.

The success of a number of specific grasp primitives in the literature reflects this observation. Although they do not lay out the implications for overall system design, they have achieved some of the most consistent functionality to date. The widespread use of guarded moves can be seen as an example of using local context to narrow the scope of variation so it can be effectively overcome, including work with parallel-jaw grippers [13], compliant hands [21, 24], and more traditional rigid hands [9]. The overhead pinch grasp used by Jain and Kemp [15] is another example, where the stereotyped action provides the ability to use "low-dimensional task-relevant features" for control. Another example is the push-grasp primitive presented by Dogar and Srinvasa [6]. In this case, sliding frictional contact is used to align a tall object in a power grasp. In this case, the specific context of the grasp primitive makes it possible to analyze the impact of friction on the motion of the object to calculate the translational displacement necessary to align the object in the hand. Kazemi et al. present a force-compliant grasping skill designed to lift small objects from flat supporting surfaces into a power grasp [16] – the context of the surface makes it easy to understand where to use compliance to correct interaction forces, and the basic idea was used by most teams in the DARPA Autonomous Robotic Manipulation Challenge [14, 26].

In all these cases, what is missing has been a good way to compare these different primitives, and a framework to understand how to create more comprehensive capabilities. It is important to note that in many cases, establishing an inner bound for variation tolerance may be sufficient–such an approximation may underestimate system performance, but will not lead to failed grasps.

In conclusion, we present a framework that uses variation as a lens to understand generality in robot grasping. First, we demonstrate that system's ability to overcome variation provides a way to compare and evaluate the capabilities of different grasping systems and apply it to a collection of leading examples. Second, we present a methodology for designing grasping systems based on the observation that *it is easier to design around local variation than to create effective parameterizations of global variation*. Analyzing variation around specific grasp configurations provides a local context that makes it tractable to create a set of *basis grasps* that span a quantifiable range of object variation. This is an important step to move from *ad hoc* approaches towards more rigorous system design and analysis.

Acknowledgements This work was supported by the National Science Foundation under Award No. IIS-0905180, and by the Defense Advanced Research Project Agency under contract number W91CRB-10-C-0141.

References

1. Robotiq adaptive gripper 3-finger model. http://robotiq.com/media/Robotiq-3-Finger-Adaptive-Robot-Gripper-Specifications.pdf (2013). Accessed 14 May 2014
2. Aukes, D., Kim, S., Garcia, P., Edsinger, A., Cutkosky, M.: Selectively compliant underactuated hand for mobile manipulation. In: 2012 IEEE International Conference on Robotics and Automation (ICRA), pp. 2824–2829 (2012). doi:10.1109/ICRA.2012.6224738
3. Bagnell, J., Cavalcanti, F., Cui, L., Galluzzo, T., Hebert, M., Kazemi, M., Klingensmith, M., Libby, J., Liu, T.Y., Pollard, N., Pivtoraiko, M., Valois, J.S., Zhu, R.: An integrated system for autonomous robotics manipulation. In: 2012 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp. 2955–2962 (2012). doi:10.1109/IROS.2012.6385888
4. Butterfass, J., Grebenstein, M., Liu, H., Hirzinger, G.: Dlr-hand ii: Next generation of a dextrous robot hand. In: IEEE International Conference on Robotics and Automation ICRA, Proceedings 2001, vol. 1, pp. 109–114. IEEE, New York (2001)
5. Diankov, R., Kuffner, J.: Openrave: A planning architecture for autonomous robotics. Robotics Institute, Pittsburgh, PA, Technical Report CMU-RI-TR-08-34 p. 79 (2008)
6. Dogar, M., Srinivasa, S.: Push-grasping with dexterous hands: mechanics and a method. In: 2010 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp. 2123–2130 (2010). doi:10.1109/IROS.2010.5652970
7. Dollar, A.M., Howe, R.D.: A robust compliant grasper via shape deposition manufacturing. IEEE/ASME Trans. Mech. **11**(2), 154–161 (2006)
8. Dollar, A.M., Howe, R.D.: The highly adaptive sdm hand: design and performance evaluation. Int. J. Robot. Res. **29**(5), 585–597 (2010)
9. Felip, J., Morales, A.: Robust sensor-based grasp primitive for a three-finger robot hand. In: 2009 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp. 1811–1816. IEEE (2009)
10. Ferrari, C., Canny, J.: Planning optimal grasps. In: IEEE International Conference on Robotics and Automation, Proceedings 1992, pp. 2290–2295. IEEE, New York (1992)
11. Hackett, D., Pippine, J., Watson, A., Sullivan, C., Pratt, G.: Foreword to the special issue on autonomous grasping and manipulation. Auton. Robot. **36**(1–2), 5–9 (2014)
12. Herzog, A., Pastor, P., Kalakrishnan, M., Righetti, L., Bohg, J., Asfour, T., Schaal, S.: Learning of grasp selection based on shape-templates. Auton. Robot. **36**(1–2), 51–65 (2014)
13. Hsiao, K., Chitta, S., Ciocarlie, M., Jones, E.: Contact-reactive grasping of objects with partial shape information. In: 2010 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp. 1228–1235 (2010). doi:10.1109/IROS.2010.5649494

14. Hudson, N., Ma, J., Hebert, P., Jain, A., Bajracharya, M., Allen, T., Sharan, R., Horowitz, M., Kuo, C., Howard, T., et al.: Model-based autonomous system for performing dexterous, human-level manipulation tasks. Auton. Robot. **36**(1–2), 31–49 (2014)
15. Jain, A., Kemp, C.C.: El-e: an assistive mobile manipulator that autonomously fetches objects from flat surfaces. Auton. Robot. **28**(1), 45–64 (2010)
16. Kazemi, M., sebastien Valois, J., Bagnell, J.A., Pollard, N.: Robust object grasping using force compliant motion primitives. In: Robotics: Science and Systems (2012)
17. Klingbeil, E., Rao, D., Carpenter, B., Ganapathi, V., Ng, A., Khatib, O.: Grasping with application to an autonomous checkout robot. In: 2011 IEEE International Conference on Robotics and Automation (ICRA), pp. 2837–2844 (2011). doi:10.1109/ICRA.2011.5980287
18. Laliberte, T., Birglen, L., Gosselin, C.: Underactuation in robotic grasping hands. Mach. Intell. Robot. Control **4**(3), 1–11 (2002)
19. Ma, R.R., Odhner, L.U., Dollar, A.M.: Dexterous manipulation with underactuated fingers: flip-and-pinch task. In: 2012 IEEE International Conference on Robotics and Automation (ICRA), pp. 3551–3552. IEEE (2012)
20. Mahmoud, R., Ueno, A., Tatsumi, S.: An assistive tele-operated anthropomorphic robot hand: Osaka city university hand ii. In: 2011 6th ACM/IEEE International Conference on Human-Robot Interaction (HRI), pp. 85–92 (2011)
21. Maldonado, A., Klank, U., Beetz, M.: Robotic grasping of unmodeled objects using time-of-flight range data and finger torque information. In: 2010 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp. 2586–2591. IEEE (2010)
22. Miller, A.T., Allen, P.K.: Graspit! a versatile simulator for robotic grasping. IEEE Robot. Autom. Mag. **11**(4), 110–122 (2004)
23. Mouri, T., Kawasaki, H., Yoshikawa, K., Takai, J., Ito, S.: Anthropomorphic robot hand: gifu hand iii. In: Proceedings International Conference ICCAS, pp. 1288–1293 (2002)
24. Natale, L., Torres-Jara, E.: A sensitive approach to grasping. In: Proceedings of the sixth international workshop on epigenetic robotics, pp. 87–94. Citeseer (2006)
25. Odhner, L., Jentoft, L.P., Claffee, M.R., Corson, N., Tenzer, Y., Ma, R.R., Buehler, M., Kohout, R., Howe, R.D., Dollar, A.M.: A compliant, underactuated hand for robust manipulation. Int. J. Robot. Res. **33**(5), 736–752 (2014)
26. Righetti, L., Kalakrishnan, M., Pastor, P., Binney, J., Kelly, J., Voorhies, R.C., Sukhatme, G.S., Schaal, S.: An autonomous manipulation system based on force control and optimization. Auton. Robot. **36**(1–2), 11–30 (2014)
27. Saxena, A., Driemeyer, J., Ng, A.Y.: Robotic grasping of novel objects using vision. Int. J. Robot. Res. **27**(2), 157–173 (2008)
28. Shimoga, K.B.: Robot grasp synthesis algorithms: a survey. Int. J. Robot. Res. **15**(3), 230–266 (1996)
29. Weisz, J., Allen, P.: Pose error robust grasping from contact wrench space metrics. In: 2012 IEEE International Conference on Robotics and Automation (ICRA), pp. 557–562 (2012). doi:10.1109/ICRA.2012.6224697

Using Fractional Order Elements for Haptic Rendering

Ozan Tokatli and Volkan Patoglu

1 Introduction

The goal of haptic rendering is to synthetically recreate virtual environments as close to reality as possible, while simultaneously ensuring safety of the interaction between the human operator and the haptic display. However, there is a well-known trade-off between stability robustness and transparency of interaction and there exists a continual search for new approaches to improve the rendering quality of the haptic systems, while ensuring coupled stability of interaction.

While the environments to be rendered can vary widely, ranging from rigid bodies to elastic materials, and even to fluids, the stability robustness has been most commonly studied for the simple environment model that consists of a linear spring and a damper. This model has been shown to capture many important aspects of haptic rendering, from the sampled-data nature of the haptic systems to the presence of the human operator in the loop.

While the classical linear elastic models can be used to capture the natural behaviour of many environments, these models fall short of capturing some other important natural phenomenon, such as time dependent stress relaxation of viscoelastic elements, a crucial aspect required to faithfully model mammal tissue. In particular, viscoelastic materials display elasticity and viscosity properties simultaneously, generalizing the existing theories for solids and viscous materials. Modeling the complex behaviour of viscoelastic materials is an active research area and it has been recognized that fractional order calculus is an effective tool to model these materials with fewer parameters and simple mathematical structures [5]. For instance, the standard linear solid (SLS) model has been shown to faithfully model human prostate tissue

O. Tokatli (✉) · V. Patoglu
Sabanci University, 34956 Orhanli, Tuzla, Turkey
e-mail: otokatli@sabanciuniv.edu

V. Patoglu
e-mail: vpatoglu@sabanciuniv.edu

A. Bicchi and W. Burgard (eds.), *Robotics Research*, Springer Proceedings in Advanced Robotics 2, DOI 10.1007/978-3-319-51532-8_23

[42], since this model is capable of capturing the time dependent creep compliance property of the tissue.

Fractional order calculus is a generalization of the familiar integer order calculus in that it allows for differentiation/integration, called differointegration, with orders of *any real number*. Intuitively, a fractional order derivative behaves as an interpolation between the neighboring integer order derivatives, due the continuous behavior of the differointegration operator with respect to its order. For instance, considering position signal as the input, continually varying the order of differentiation order from 1 to 0 acts as changing the properties of a linear mechanical element from a pure dissipation element towards a pure potential energy storage (stiffness) element. Likewise, tuning the differentiation order from 1 to 2 acts as continually transforming from a pure dissipation element towards a pure kinetic energy storage (inertia) element. Note that dissipation exits for all differentiation orders in the open interval (0, 2), while pure energy storage takes place only for the integer orders.

Inspired by the existence of fractional order models in the nature, we propose the use of fractional order models/controllers in haptic systems. We generalize the existing results based on linear elastic and viscous mechanical elements to models with linear fractional order elements. The fractional order model not only can recover the classical virtual environment model of consisting of springs and dampers, but also enable rendering of realistic viscoelastic materials thanks to the fractional order differointegration term in its model.

The use of fractional order calculus in systems and control applications is known to provide the user with an extra parameter, the order of differointegration, which can be tuned to improve the desired behaviour of the overall system. This property of fractional order controllers is widely employed for robust motion control applications. For haptic systems, introducing a proper amount of dissipation is essential for achieving coupled stability, as well as improving their transient response during interactions. However, dissipation can adversely affect the transparency of the rendering by distorting the match between the desired and rendered impedance values. Fractional calculus based control is a promising generalization in that it provides an alternative means for tuning the characteristics of the dissipation supplied to the system, through the adjustment of the order of differentiation. In particular, since the fractional calculus generalization provides an additional degree of freedom for adjusting the dissipation behaviour of the overall system, fractional order haptic rendering has the potential to improve upon the stability robustness-transparency trade-off dictated by the integer order analysis.

Along these lines, we study haptic rendering of fractional order impedances and explore how the use of fractional order elements impacts the coupled stability of the overall sampled-data system. Our results generalize the well-known passivity condition to include fractional order impedances and demonstrate the effect of the order of differointegration on the passivity boundary. We also characterize the effective stiffness and damping behavior of the fractional order impedance as a function of frequency and differointegration order. Even though there has been an investigation of haptic rendering of viscoelastic materials in [25], passivity and effective impedance analysis of fractional order models have not been studied to the best of authors' knowledge.

The rest of the paper is organized as follows: In Sect. 2, we review the literature on the coupled stability of haptic systems (Sect. 2.1) and the fractional order control (Sect. 2.2). In Sect. 3, we provide preliminaries on the analysis of haptic and fractional order systems. The main results of the paper are given in Sect. 4, while their implications are discussed in Sect. 5. Finally, Sect. 6 concludes the paper.

2 Related Work

2.1 Coupled Stability of Haptic Systems

A haptic system is desired to stay stable at all times, for any human operator, and under any operation/grip conditions. The presence of the human operator in the loop significantly complicates the coupled stability analysis and controller design of haptic systems. The first and the foremost challenge is to find a simple and reliable model for the human operator. Without a model of the human operator, determining the coupled stability of the haptic system is not a trivial task. Furthermore, the sampled-data nature of the haptic systems introduces an extra challenge to the analysis.

The coupled stability analysis of haptic systems can be loosely categorized into two different approaches. The first approach assumes a model for the human operator and checks for the overall stability of the system based on this model. On the other hand, the second approach focuses on the haptic system alone and aims at robust stability of the haptic system for a certain, but wide, range of human operator models. Despite the conservative nature of the results obtained, the latter approach is widely accepted as a more robust approach in designing haptic controllers.

The early literature commonly adopted modeling the human operator approach. In these works, researchers have assumed simple, typically 2nd order linear time invariant (LTI), models representing the human operator in the loop. The pioneering work on the haptic system stability has been presented by Minsky et al. [33], where the human operator is approximated with a second order LTI model, while the discrete elements, the sampler and the hold, are approximated with continuous time models. Nyquist stability criterion is used to determine the stability of the overall system. In [19], Gillespie et al. adopted a similar approach and has shown that, the switching nature of the virtual wall and its discrete time implementation causes energy leaks and this leakage may cause instability. Stability analysis methods have also been used to study the effects of various different aspects of a haptic system. In particular, in [18], Routh–Hurwitz criteria is used to characterize the uncoupled stability of the haptic system, where the human operator is not attached to the robot. A similar analysis is conducted in [23] to investigate the effects of physical damping, time delay, human operator on uncoupled stability. A Lyapunov based approach is introduced in [15] to determine the stability of the haptic system and the effects of discretization, quantization, time delay and Coulomb friction on the stability of haptic systems are explored.

Most of the studies on coupled stability analysis rely on methods that do not require a specific model for the human operator. In this approach, stability is considered for all possible human models within a certain class; therefore, the results obtained are generally more robust, but also more conservative, compared to human model based analysis. The pioneering method in this branch is based on passivity analysis. The passivity based methods assume that the human operator is a passive element. This widely accepted assumption is based on [22], where it has been shown that within the frequency range required for the haptic applications, humans generally act as passive network elements. In the teleoperation field, passivity analysis is first applied by Anderson and Spong [2] on continuous time models. Later, in his seminal work, Colgate introduced the passivity theorem for sampled-data system and applied it to haptics [9]. Without the need for a human model, the overall haptic system is rendered passive, so that any possible instabilities that might occur are avoided. Colgate's theorem had a profound impact in the field, since it handles the haptic system as a sampled-data system. Later, the passivity approach is generalized for interactions with unknown passive virtual environments through the idea of virtual coupling [11]. Virtual coupling acts as a buffer between the virtual environment and the robot; moreover, the parameters of the coupler are selected such that the robot-coupler 2-port network is passive. The virtual coupler idea is further extended in [1] with unconditional stability theorem for 2-port networks and to include admittance type devices.

Several other approaches exist to ensure coupled stability of haptic interactions. The time domain passivity approach by Hannaford and Ryu [21, 40] along with its variations [24] and the bounded impedance analysis by Haddadi and Hastrudi-Zaad [20] are among the most notable ones.

2.2 *Fractional Order Control*

Fractional order calculus has its roots in the late 17th century, in other words, it is a peer of the celebrated calculus with integer order derivatives and integrals. Some argue that the idea of fractional order calculus can be found in the letters of L'Hopital, Leibniz and Bernoulli. However, the true development of fractional order calculus falls into the 18th and 19th centuries.

The idea of differentiating/integrating a function with an arbitrary order is, at first sight, counter-intuitve, but it is a phenomenon that has been observed in nature. For instance, modeling viscoelastic materials using fractional order calculus is known to yield better results than modeling these materials with integer order calculus. In [3], it has been shown that using fractional order models can significantly reduce the degree of the model. In [13], a fractional order model is proposed for red blood cells.

A fractional differointegral, fills the gap between the consecutive integer order derivatives/integrals. In other words, in terms of the differentiation/integration, derivative/integral operator is not discrete; instead, there is a smooth transition between successive integer orders of differentiation/integration.

Fractional order control has also found wide applications in the field of robotics and controls. A fractional order controller, called CRONE, can be designed to exhibit iso-damping behaviour, even if the parameters of the system are changed significantly [37]. This control method has been successfully implemented in industrial applications, such as in car suspensions. A fractional order counterpart of the reputable PID controller is also commonly employed in motion control applications [39]. Tilted integral derivative (TID) is another commonly used fractional calculus based controller [30]. A quantitative comparison of these controllers with respect to their integer order counterparts can be found in [41]. Fractional order controllers are mostly used robust motion control [31]; however, they have been also applied to position-force hybrid control in [17].

There exists many notable books covering fractional order control [4, 29, 34, 35, 38], as well as several tutorial/review papers [7, 16, 26, 27, 32].

From the point of view of this paper, the arbitrary differentiation order can be interpreted as an alternative means of adjusting impedance and dissipation characteristics of the overall haptic system. To improve the stability robustness of haptic devices, the conventional approach is to introduce viscous damping, that is proportional to the first derivative of the position signal. However, with a fractional order control approach, the dissipation can be supplied to the system with arbitrary differentiation order. The order of differentiation defines the dissipation as well as other mechanical impedance characteristics of the fractional order mechanical element. In particular, a differentiation order less than 1 results in a dissipative mechanical element that can store potential energy, while a differentiation order greater than 1 is a dissipative element that can also store kinetic energy [28].

3 Preliminaries

The symbol s indicates that a transfer function is in continuous time domain, whereas z indicates a discrete time transfer function. Subscript h denotes a human operator, d refers to the haptic device and e signifies an environment.

3.1 The Haptic System

Figure 1 presents the block diagram of the haptic system in a sampled-data form. The human is represented with the model ϕ, which is possibly nonlinear and assumed to be passive. $G(s)$ denotes the haptic display with m and b denoting the mass and physical viscous damping of the robot. The feedback signal is chosen as the position of the robot and it is sampled with a time period of T. $H(z)$ represents the model of the virtual wall that is implemented on a digital computer. Finally, the computed

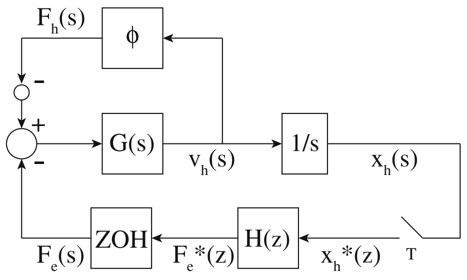


Fig. 1 The sampled-data haptic system with ideal sampler and zero order hold

reaction force, F_e^* is passed through a zero-order hold and fed back to the plant. The following two subsections elaborate on the components of this sampled-data system.

3.2 The Rigid Body Model of the Robot

The haptic interface is modeled as a rigid robot and it is assumed that the human operator firmly grasps the robot; hence, $x_h = x_d$. The equations of motion for this system can be given as

$$m\ddot{x}_h + b\dot{x}_h = f_h + f_e \tag{1}$$

The corresponding transfer function of the haptic display from force to velocity in continuous time is

$$G(s) = \frac{1}{ms + b} \tag{2}$$

3.3 The Virtual Environment

The aim of this paper is to investigate the effect of a fractional order elements in the virtual environment. Therefore, a simple virtual wall model consisting of a spring and a fractional order linear element is considered.

$$H(z) = K + B\left(\frac{1 - z^{-1}}{T}\right)^{\alpha} \tag{3}$$

In this virtual wall model, K and B are respectively the virtual stiffness and the fractional order linear element parameters of the virtual environment. Throughout the analysis, the velocity of the robot is approximated using the backward difference method. Although different velocity approximation methods can be employed for discrete time implementations of the virtual wall, the finite difference approach is chosen due to its simplicity. Moreover, since this approach has been extensively used in the literature, this choice enables a comparison of the performance for integer and fractional order models. It is important to note that in this virtual wall model, the order of the differentiator is not necessarily an integer number. In this paper, we conduct the analysis for $\alpha \in [0, 2]$, where $\alpha = 1$ corresponds to the classical backward difference differentiator. We also consider positive values of K and B values for simplicity.

For the upcoming analysis, it is necessary to introduce nondimensional forms of the system parameters, because as the differentiation order changes, the physical meaning, as well as the unit of B changes. Nondimensional parameters enable comparison of virtual walls with different differential orders. The nondimensionalization is achieved through the following transformations. Note that these transformations are similar to ones noted in [23], except for the dissipation element of the virtual environment. Here, we have generalize the nondimensionalization of virtual damping to fractional order dissipative elements.

$$K \rightarrow \kappa = \frac{KT^2}{m}, \quad B \rightarrow \beta = \frac{BT^{2-\alpha}}{m}, \quad b \rightarrow \delta = \frac{bT}{m} \tag{4}$$

3.4 Passivity of the Haptic Interface

The passivity analysis of a haptic display is first analyzed in [10]. Later in [8] the passivity condition is formalized. Since the analysis of this paper relies on this theorem, the theorem is repeated from [9].

Theorem 1 (Passivity of a haptic interface [9]) *A necessary and sufficient condition for the passivity of the haptic interface model in Fig. 1 is*

$$b > \frac{T}{2}\frac{1}{1-\cos(\omega T)}Re\left\{\left(1-e^{-j\omega T}\right)H(e^{j\omega T}))\right\} \tag{5}$$

for $0 \le \omega \le \omega_N$, *where* $\omega_N = \pi/T$ *is the Nyquist frequency.*

One of the important aspects of this theorem is that it analyzes a haptic system as a sampled-data system. Hence, the adverse effects of the sample and hold elements are taken into consideration.

3.5 Fractional Order Calculus

In order to familiarize the concept of fraction order calculus and understand key concepts, this section briefly introduces fractional order differointegration.

In the literature, many definitions of fractional order differointegrals exits. The most frequently used definitions are the Grunwald–Letnikov, Riemann–Liouville and Caputo's definitions. Riemann–Liouville is the most frequently used definition and is defined as

$$ {}_a\mathscr{D}_t^{\alpha} f(t) = \frac{1}{\Gamma(n-\alpha)} \frac{d^n}{dt^n} \int_a^t \frac{f(\tau)}{(t-\tau)^{\alpha-n+1}} d\tau $$

for $n-1<\alpha<n$, where Γ represents the gamma function.

Grunwald–Letnikov differointegral definition is important since it forms a basis for the discrete implementation

$$ {}_a\mathscr{D}_t^{\alpha} f(t) = \lim_{h\to\infty} h^{-\alpha} \sum_{j=0}^{[\frac{t-a}{h}]} (-1)^j \binom{\alpha}{j} f(t-jh) $$

where [.] indicates the integer part of the real number.

Despite the frequent use of the previous definitions, in control systems, Caputo's definition is preferred, since handling of the initial conditions is more intuitive with this definition.

$$ {}_a\mathscr{D}_t^{\alpha} f(t) = \frac{1}{\Gamma(n-\alpha)} \int_a^t \frac{f^{(n)}(\tau)}{(t-\tau)^{\alpha-n+1}} d\tau $$

for $n-1<\alpha<n$. The analysis in this paper implicitly uses the Caputo's definition, due to its advantages in terms of computing the Laplace transformation of functions with fractional order differointegrals.

The properties of fractional order differointegral operator is summarized in the following list.

- The fractional order differointegral is a linear operator.

$$ {}_a\mathscr{D}_t^{\alpha}(f(t)+g(t)) = {}_a\mathscr{D}_t^{\alpha} f(t) + {}_a\mathscr{D}_t^{\alpha} g(t) $$

- The fractional differointegral operator is causal. Assume $f(t)=0$ for $t<0$, then ${}_a\mathscr{D}_t^{\alpha} f(t) = 0$.
- The fractional differointegral operator is shift invariant.

$$ {}_a\mathscr{D}_t^{\alpha} f(t-t_0) = {}_a\mathscr{D}_t^{\alpha} f(\tau)|_{\tau=t-t_0} $$

- If $f(t)$ is an analytic function of t, then its derivative is also analytic in both t and α, where α is the order of differentiation.
- For $\alpha \in \mathbb{Z}$, the result of the fractional order derivative operator is same as the integer order one.

- Fractional order differointegral operator has semi-group property.

$$ {}_a\mathscr{D}_t^{\alpha} f(t)\ {}_a\mathscr{D}_t^{\beta} f(t) = {}_a\mathscr{D}_t^{\beta} f(t)\ {}_a\mathscr{D}_t^{\alpha} f(t) = {}_a\mathscr{D}_t^{\alpha+\beta} f(t) $$

- Using the Caputo's definition, the Laplace transform is defined as

$$ \mathscr{L}({}_a\mathscr{D}_t^{\alpha} f(t)) = s^{\alpha} \mathscr{L}(f(t)) $$

Note that, according to all definitions, differointegration is a nonlocal phenomena and is history-dependent. However, one can observe from the coefficients in Grunwald–Letnikov definition that for large values of t, the role of the history of the behaviour of the function $f(t)$ near the initial condition $(t = a)$ can be neglected under mild conditions. This observation serves as the basis of the *short memory principle*, where one takes into account the behavior of $f(t)$ only in the recent past, that is, in the interval $[t - L,\ t]$, where L is defined as the memory length. According to short memory principle, fractional order differointegration with initial condition at a is approximated with the fractional order differointegration with moving initial condition at $t - L$, where the desired level of accuracy with this approximation can be achieved by adjusting the memory.

There exist two main approaches for discretization of fractional order differointegration operation. The first approach is direct discretization, where the exact mathematical model of the fractional order differointegral is used for further analysis. These direct discretization methods generally consider series expansions, such as MacLaurin series expansion, power series expansion, and continued fraction expansion. In [6], direct discretization is analyzed and polynomial approximations for arbitrary order differointegration is introduced. For the indirect method, a mathematical model is fitted to the frequency domain response of the fractional order differointegral. Details of different discretization schemes can be found in [14].

For further details of fractional order calculus, the reader is referred to [7, 34, 36]. After understanding the haptic system and familiarizing with the fractional order calculus, we can, now proceed to the results of this paper.

4 Results

4.1 Passivity Analysis

Corollary 1 *Consider a haptic system with a robot model as given in Eq. 2 and a virtual environment model as described in Eq. 3 inside the control architecture introduced in Fig. 1, where human is modeled as passive operator. For positive values of B and K, the overall system is passive if the following inequality is satisfied.*

$$b > \frac{KT}{2} + B\left(\frac{T}{2}\right)^{1-\alpha} \tag{6}$$

The dimensionless form of Eq. 6 can be expressed as

$$\delta > \frac{\kappa}{2} + \beta\left(\frac{1}{2}\right)^{1-\alpha} \tag{7}$$

where non-dimensionalization is performed according to Eq. 4.

Proof Corollary 1 follows Theorem 1 in Sect. 3.4. In particular, let the brach cut for the analysis be chosen at $-\pi$ and consider the first Riemannian sheet, which is physically meaningful. Replace the virtual wall model of Eq. 5 with the virtual wall model of Eq. 3 to obtain

$$b > \frac{T/2}{1-\cos(\omega T)} \Re\left\{\left(1-e^{-j\omega T}\right)\left(K + B\left(\frac{1-e^{-j\omega T}}{T}\right)^{\alpha}\right)\right\} \tag{8}$$

$$b > \frac{KT}{2} + \frac{BT^{1-\alpha}}{2}\frac{\Re\left\{\left(1-e^{-j\omega T}\right)^{1+\alpha}\right\}}{1-\cos(\omega T)} \tag{9}$$

Representing $1-e^{-j\omega T}$ in the phasor notation and substituting for $1-\cos\omega T$

$$1-e^{-j\omega T} = \sqrt{2(1-\cos\omega T)}\, e^{-j\frac{\omega T-\pi}{2}} \tag{10}$$

$$1-\cos\omega T = 2\sin^2\frac{\omega T}{2} \tag{11}$$

one can further manipulate the equations as follows

$$b > \frac{KT}{2} + \frac{BT^{1-\alpha}}{2}\frac{\Re\left\{\left(\sqrt{2(1-\cos\omega T)}\, e^{-j\frac{\omega T-\pi}{2}}\right)^{1+\alpha}\right\}}{2\sin^2\frac{\omega T}{2}} \tag{12}$$

$$b > \frac{KT}{2} + \frac{BT^{1-\alpha}}{2}\frac{\left(\sqrt{4\sin^2\frac{\omega T}{2}}\right)^{1+\alpha}\Re\left\{e^{-j\frac{\omega T-\pi}{2}(1+\alpha)}\right\}}{2\sin^2\frac{\omega T}{2}} \tag{13}$$

$$b > \frac{KT}{2} + \frac{BT^{1-\alpha}}{2}\frac{\left(2\sin\frac{\omega T}{2}\right)^{1+\alpha}\cos\left(\frac{\omega T-\pi}{2}(1+\alpha)\right)}{2\sin^2\frac{\omega T}{2}} \tag{14}$$

$$b > \frac{KT}{2} + B\left(\frac{T}{2}\right)^{1-\alpha}\left(\sin\frac{\omega T}{2}\right)^{\alpha-1}\cos\left(\frac{\omega T-\pi}{2}(1+\alpha)\right) \tag{15}$$

The system is passive if Eq. 15 holds for all frequencies $0 \le \omega \le \pi/T$. In order to obtain Eq. 6, the worst-case scenario, or the maximum value of the frequency dependent part of the previous inequality, has to be determined, since B is known to be positive. Let the frequency dependent part of the inequality be represented as

$$f(\omega,\alpha) = \left(\sin\frac{\omega T}{2}\right)^{\alpha-1} \cos\left(\frac{\omega T-\pi}{2}(1+\alpha)\right) \tag{16}$$

The extremum of this function occurs at frequencies where $\partial f(\omega,\alpha)/\partial\omega = 0$. The first partial derivative of $f(\omega,\alpha)$ with respect to ω can be expressed as

$$\begin{aligned}\frac{\partial f(\omega,\alpha)}{\partial\omega} = -\left(\sin\frac{\omega T}{2}\right)^{\alpha-1} \Bigg[&(1-\alpha)\cot\frac{\omega T}{2}\cos\left(\frac{\omega T-\pi}{2}(1+\alpha)\right)\dots \\ &+ (1+\alpha)\sin\left(\frac{\omega T-\pi}{2}(1+\alpha)\right)\Bigg]\end{aligned} \tag{17}$$

After some manipulations, Eq. 17 can be transformed into

$$\sin\left(\frac{\omega T-\pi}{2}\right) - \alpha\sin\left(\omega T - \frac{\omega T-\pi}{2}\alpha\right) = 0 \tag{18}$$

For an arbitrary α, this equation holds if both sine terms vanish and this condition occurs at $\omega = \pi/T$. Moreover, the second partial derivative of $f(\omega,\alpha)$ with respect ω to is negative for $\omega = \pi/T$, ensuring that $\omega = \pi/T$ is where the function attains a maximum value. Closely investigating the 3D plot of $f(\omega,\alpha)$ confirms that the global maximum is always attained at $\omega = \pi/T$, the Nyquist frequency of the sampled-data system. Substituting this value into Eq. 15 completes the proof.

Remark 1 Besides from the usual virtual wall parameters, K and B, fractional order controller introduces a new design parameter, α, which can be set to any real number. The new parameter explicitly shows up in the passivity condition and introduces new opportunities to improve the overall performance of the haptic system. Figure 2 depicts the solution of Eq. 6 for various values of α.

Remark 2 Equation 6 is a generalization of the celebrated passivity condition for haptic systems, introduced by Colgate in [9], to the fractional order case. A close investigation reveals that, for $\alpha = 1$, Eq. 6 can recover the familiar integer order condition. Moreover, the other integer order cases of $\alpha = \{0, 2\}$ can also be easily recovered from Eq. 6.

4.2 Effective Impedance of the Fractional Order Virtual Wall

Section 4.1 analyses the effect of using a fractional order controller for the virtual environment on the passivity characteristics of the haptic device. In this section, we investigate the effect of fractional order models on transparency of haptic rendering by studying the effective impedance of the virtual environments as a function of

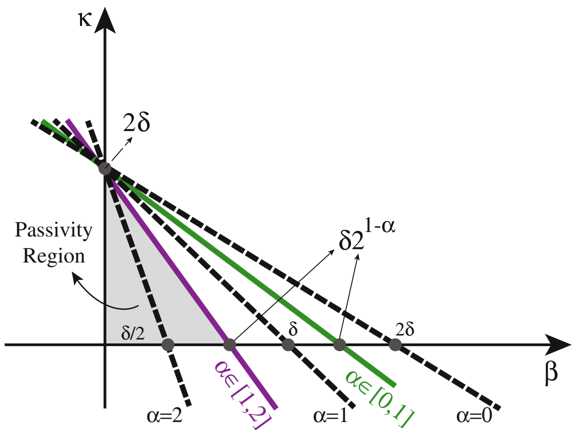


Fig. 2 Nondimensional passivity regions for different values of differentiation order α

input frequency, as proposed in [12]. This analysis not only reveals how a fractional order virtual wall behaves in the frequency range up to the Nyquist frequency, but also may help decide on a proper differentiation order for a given task.

In order to perform the effective impedance analysis on the virtual environment with fractional order model, the definitions of effective stiffness (ES) and effective damping (ED) are adjusted for position feedback as

$$ES(\omega) = \Re^{+}\{H(e^{j\omega T})\} \tag{19}$$

$$ED(\omega) = \frac{1}{\omega}\Im^{+}\{H(e^{j\omega T})\} \tag{20}$$

For the virtual wall model given in Eq. 3, the effective stiffness and damping are read as

$$ES(\omega) = K + B\left(2\sin\frac{\omega T}{2}\right)^{\alpha}\cos\left(\frac{\omega T-\pi}{2}\alpha\right) \tag{21}$$

$$ED(\omega) = \quad -B\left(2\sin\frac{\omega T}{2}\right)^{\alpha}\sin\left(\frac{\omega T-\pi}{2}\alpha\right) \tag{22}$$

Note that $-\pi/2 \le (\omega T - \pi)/2 \le 0$ lives in the fourth quadrant; hence, for $0 \le \alpha \le 1$, $(\omega T - \pi)\alpha/2$ is always in the fourth quadrant, while for $1 \le \alpha \le 2$, $(\omega T - \pi)\alpha/2$ can lie in the third or the fourth quadrants.

5 Discussion

Calculus with integer order differointegrals has proved its ability to model the physical phenomena; but it is not the ultimate tool to model nature. In fact, fractional order calculus is an effective tool that broadens the modeling boundaries of the familiar calculus. Our proposition of using fractional calculus in haptics enables a new potential of rendering unorthodox impedances, such as viscoelastic materials that exhibit frequency dependent stiffness and damping characteristics within a single mechanical element. Even though approximate models for such materials with integer order differointegrals may exist, fractional order calculus is known to result in simpler and more capable models, capturing the true nature of such materials. Consequently, the use of fractional order calculus in haptics significantly extends the type of impedances that can be rendered using the integer order models.

Inclusion of fractional order models/controllers into the human-in-the-loop sampled data control loop has a direct consequence on the coupled stability characteristics of the overall system. In particular, Eq. 7 generalizes the well known passivity condition, $\delta > \frac{\kappa}{2} + \beta$ in the nondimensional form, to include factional order models. An important observation from this equation is the fact that the size of dimensionless κ-β passivity region can be modulated by tuning the order of the differointegral. Figure 2 provides a visual demonstration of this result, where $\alpha = 1$ represents the virtual wall with integer order damping term. For $\alpha \in [0, 1)$, the fractional order differointegral term increases the nondimensional area of the κ-β passivity region. The minimum passivity region occurs as $\alpha \to 2$, where the fractional order element acts as a kinetic energy storage element (inertia).

In Sect. 4.2 we have presented expressions for calculating effective impedance of discrete time fractional virtual environments. This frequency dependent analysis reveals that a fractional order element between consecutive integer orders inherits the mechanical properties corresponding to those integer orders. Figure 3 depicts the three frequency dependent coefficients in Eqs. 21 and 22 that shape the response of effective spring and damping terms.

In Eq. 21 characterizing the effective stiffness, κ is a positive number, always contributing positively to the effective stiffness. On the other hand, effective stiffness also has a β dependent term that can increase or decrease its value as a continuous function of ω and α. If $0 \leq \alpha \leq 1$, cosine term in Eq. 21 is always positive, independent of ω; hence, the contribution of β on the effective stiffness is always positive. However, if $1 \leq \alpha \leq 2$, then the cosine term can change sign; therefore, depending of the frequency, the effective stiffness can also be lowered.

In Eq. 22 characterizing the effective damping, as expected, one can observe that there is no contribution of κ. Effective damping should always be positive, and this is indeed the case, since $\sin\left(\frac{\omega T - \pi}{2}\alpha\right)$ is always in third or fourth quadrants. As a result for $\alpha \in (0, 2)$, the effective damping is positive and there is dissipation in the system. The magnitude of the effective damping is predominantly determined by $(2\sin(\omega T/2))^{\alpha}$ term and by choosing $1 \leq \alpha \leq 2$ the effective damping can be increased significantly at high frequencies, compared to $0 \leq \alpha \leq 1$.

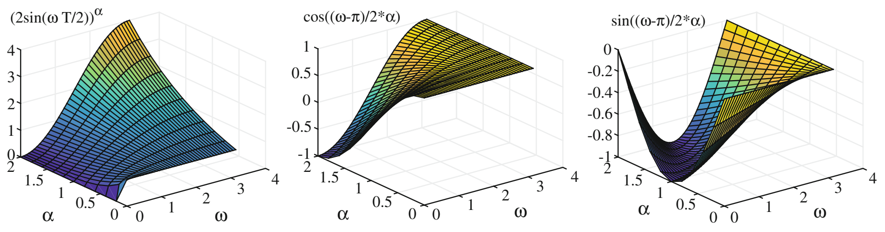


Fig. 3 Coefficients that scale effective stiffness and damping of the fractional order virtual environment

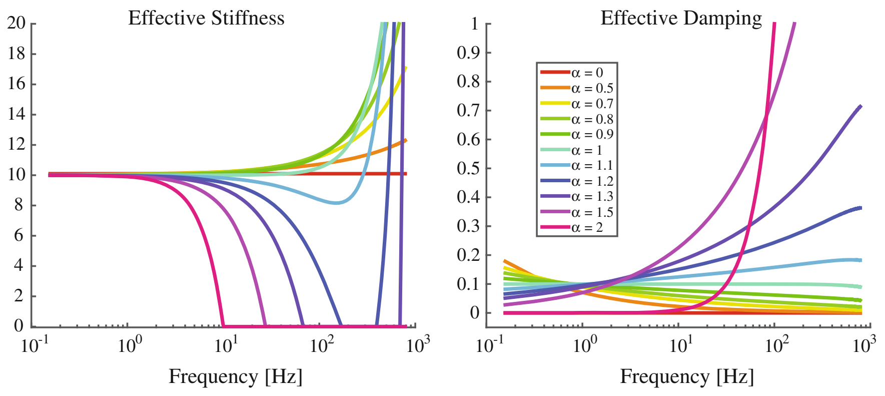


Fig. 4 Effective stiffness and damping of the fractional order virtual environment of Eq. 3

Figure 4 depicts the effective stiffness and damping of a sample fractional order virtual environment with $K = 10$ N/mm, $B = 0.1$ Ns/mm and $T = 0.001$ s. From the figure, the frequency and differointegration order dependance of the effective stiffness and damping can be observed. Noting the frequency separation between human input and noise, differointegration order can be put in good use to adjust the frequency characteristics of effective impedance such that good transparency behavior can be ensured within the human bandwidth, while better stability robustness is achieved at higher frequencies.

6 Conclusions

We have proposed using fractional order models/controllers for haptic rendering and explored the impact of fractional order elements to the coupled stability of the overall sampled-data system. We also characterized the effective stiffness and damping behavior of the fractional order impedance as a function of frequency and differointegration order.

References

1. Adams, R.J., Hannaford, B.: Stable haptic interaction with virtual environments. IEEE Trans. Robot. Autom. **15**(3), 465–474 (1999)
2. Anderson, R., Spong, M.: Bilateral control of teleoperators with time delay. IEEE Trans. Autom. Control **34**(5), 494–501 (1989)
3. Bagley, R.L., Torvik, P.J.: Fractional calculus - a different approach to the anaylsis of viscoelastically damped structures. AIAA J. **21**, 741–748 (1983)
4. Caponetta, R., Dongola, G., Fortuna, L., Petras, I.: Fractional Order Systems. World Scientific, Singapore (2010)
5. Carpinteri, A., Mainardi, F.: Fractals and Fractional Calculus in Continuum Mechanics. Springer, Wien (1997)
6. Chen, Y.Q., Moore, K.: Discretization schemes for fractional-order differentiators and integrators. IEEE Trans. Circuits Syst. I: Fundam. Theory Appl. **49**(3), 363–367 (2002)
7. Chen, Y.Q., Petras, I., Xue, D.: Fractional order control - a tutorial. In: American Control Conference, pp. 1397–1411 (2009)
8. Colgate, J., Brown, J.: Factors affecting the z-width of a haptic display. In: IEEE International Conference on Robotics and Automation, vol. 4, pp. 3205–3210 (1994)
9. Colgate, J.E., Schenkel, G.G.: Passivity of a class of sampled-data systems: application to haptic interfaces. J. Robot. Syst. **14**(1), 37–47 (1997)
10. Colgate, J., Grafing, P., Stanley, M., Schenkel, G.: Implementation of stiff virtual walls in force-reflecting interfaces. In: Virtual Reality Annual International Symposium, pp. 202–208 (1993)
11. Colgate, J., Stanley, M., Brown, J.: Issues in the haptic display of tool use. In: IEEE/RSJ International Conference on Intelligent Robots and Systems. Human Robot Interaction and Cooperative Robots, vol. 3, pp. 140–145 (1995)
12. Colonnese, N., Sketch, S., Okamura, A.: Closed-loop stiffness and damping accuracy of impedance-type haptic displays. In: IEEE Haptics Symposium (HAPTICS), pp. 97–102 (2014)
13. Craiem, D., Magin, R.L.: Fractional order models of viscoelasticity as an alternative in the analysis of red blood cell (RBC) membrane mechanics. Phys. Biol. **7**(1), 13001 (2010)
14. Das, S., Pan, I.: Fractional Order Signal Processing: Introductory Concepts and Applications. Springer, Heidelberg (2012)
15. Diolaiti, N., Niemeyer, G., Barbagli, F., Salisbury, J.: Stability of haptic rendering: discretization, quantization, time delay, and coulomb effects. IEEE Trans. Robot. **22**(2), 256–268 (2006)
16. Efe, M.: Fractional order systems in industrial automation 2014; a survey. IEEE Trans. Ind. Inf. **7**(4), 582–591 (2011)
17. Ferreira, N.M.F., Machado, J.A.T.: Fractional-order hybrid control of robotic manipulators. In: International Conference on Advanced Robotics (2003)
18. Gil, J., Avello, A., Rubio, A., Florez, J.: Stability analysis of a 1 DOF haptic interface using the Routh–Hurwitz criterion. IEEE Trans. Control Syst. Technol. **12**(4), 583–588 (2004)
19. Gillespie, R.B., Cutkosky, M.R.: Stable user-specific haptic rendering of the virtual wall. In: Proceedings of The International Mechanical Engineering Congress and Exhibition (1995)
20. Haddadi, A., Hashtrudi-Zaad, K.: Bounded-impedance absolute stability of bilateral teleoperation control systems. IEEE Trans. Haptics **3**(1), 15–27 (2010)
21. Hannaford, B., Ryu, J.H.: Time-domain passivity control of haptic interfaces. IEEE Trans. Robot. Autom. **18**(1), 1–10 (2002)
22. Hogan, N.: Controlling impedance at the man/machine interface. In: IEEE International Conference on Robotics and Automation, pp. 1626–1631 (1989)
23. Hulin, T., Preusche, C., Hirzinger, G.: Stability boundary for haptic rendering: influence of physical damping. In: IEEE/RSJ International Conference on Intelligent Robots and Systems, pp. 1570–1575 (2006)
24. Kim, J.P., Ryu, J.: Robustly stable haptic interaction control using an energy-bounding algorithm. Int. J. Robot. Res. (2009)

25. Kobayashi, Y., Moreira, P., Liu, C., Poignet, P., Zemiti, N., Fujie, M.: Haptic feedback control in medical robots through fractional viscoelastic tissue model. In: Annual International Conference of the IEEE Engineering in Medicine and Biology Society, pp. 6704–6708 (2011)
26. Krishna, B.: Studies on fractional order differentiators and integrators: a survey. Signal Process. **91**(3), 386–426 (2011)
27. Li, C., Zhang, F.: A survey on the stability of fractional differential equations. Eur. Phys. J. Spec. Top. **193**(1), 27–47 (2011)
28. Lorenzo, C.F., Hartley, T.T.: Energy considerations for mechanical fractional-order elements. J. Comput. Nonlinear Dyn. **10**, (2015)
29. Luo, Y., Chen, Y.Q.: Fractional Order Motion Controls. Wiley, New Jersey (2012)
30. Lurie, B.J.: Three-parameter tunable tilt-integral-derivative (TID) controller (1994)
31. Ma, C., Hori, Y.: Fractional-order control: theory and applications in motion control [past and present]. IEEE Ind. Electron. Mag. **1**(4), 6–16 (2007)
32. Machado, J.T., Kiryakova, V., Mainardi, F.: Recent history of fractional calculus. Commun. Nonlinear Sci. Numer. Simul. **16**(3), 1140–1153 (2011)
33. Minsky, M., Ming, O.y., Steele, O., Brooks Jr., F.P., Behensky, M.: Feeling and seeing: issues in force display. In: Proceedings of the Symposium on Interactive 3D Graphics, pp. 235–241 (1990)
34. Monje, C.A., Chen, Y.Q., Vinagre, B.M., Xue, D., Feliu-Batlle, V.: Fractional-order systems and controls: fundamentals and applications. Springer, London (2010)
35. Oldham, K.B., Spanier, J.: The Fractional Calculus. Academic Press, Cambridge (1974)
36. Ortigueira, M.D.: Fractional Calculus for Scientists and Engineers. Springer, Netherlands (2011)
37. Oustaloup, A., Mathieu, B., Lanusse, P.: The crone control of resonant plants: application to a flexible transmission. Eur. J. Control **1**(2), 113–121 (1995)
38. Petras, I.: Fractional-Order Nonlinear Systems. Springer, Heidelberg (2011)
39. Podlubny, I.: Fractional-order systems and pi/sup /spl lambda//d/sup /spl mu//-controllers. IEEE Trans. Autom. Control **44**(1), 208–214 (1999)
40. Ryu, J.H., Kwon, D.S., Hannaford, B.: Stable teleoperation with time-domain passivity control. IEEE Trans. Robot. Autom. **20**(2), 365–373 (2004)
41. Xue, D., Chen, Y.Q.: A comparative introduction of four fractional order controllers. In: 4th World Congress on Intelligent Control and Automation, vol. 4, pp. 3228–3235 (2002)
42. Zhang, M., Nigwekar, P., Castaneda, B., Hoyt, K., Joseph, J.V., di Sant'Agnese, A., Messing, E.M., Strang, J.G., Rubens, D.J., Parker, K.J.: Quantitative characterization of viscoelastic properties of human prostate correlated with histology. Ultrasound Med. Biol. **34**(7), 1033–1042 (2008)

Design of a Novel 3-DoF Serial-Parallel Robotic Wrist: A Symmetric Space Approach

Yuanqing Wu and Marco Carricato

1 Introduction

Robot wrist design is a key technology in developing robot manipulators for dexterous manipulation. A comprehensive review can be found in [1]. Two common kinematic designs of serial wrists are the ZYZ (roll-pitch-roll) and the XYZ (pitch-yaw-roll) axis configurations [1]. The wrist orientation range can be characterized by a pointing cone of the unit sphere S^2 depicting the admissible directions of the last roll axis [2]. Mathematically, this corresponds to a characterization of the special orthogonal group SO(3) as a fibre bundle over S^2, with typical fibre SO(2). A modified Euler angle parametrization, relying on tilt and torsion angles, is proposed in [3], which exactly reflects this topological model (see Fig. 1). So far, serial wrists remain dominant in industrial robotics. However, for topological reasons [4], any (non-redundant) serial wrist necessarily admits inverse kinematics singularities, or stationary configurations [5], where the three axes of the wrist become coplanar [2]. Singular or near-singular configurations cause a rapid increase in joint rates and should be avoided [6]. Consequently, the singularity-free pointing cone of a serial wrist is usually restricted to 150° [2]. Besides, it is reported that the last roll axis may suffer from a decreased stiffness due to backlash present in the transmission gear train [7].

Parallel kinematics wrists may have improved stiffness, but aggravated singularity problems, and a smaller orientation range. Both are due to the presence of loop constraints. Therefore, parallel wrists are only used in high-speed applications, when a large orientation range is not required [8, 9]. The Agile Eye [8], for example,

Y. Wu (✉) · M. Carricato
University of Bologna, Bologna, Italy
e-mail: yuanqing.wu@unibo.it; troy.woo@gmail.com

M. Carricato
e-mail: marco.carricato@unibo.it

A. Bicchi and W. Burgard (eds.), *Robotics Research*, Springer Proceedings in Advanced Robotics 2, DOI 10.1007/978-3-319-51532-8_24

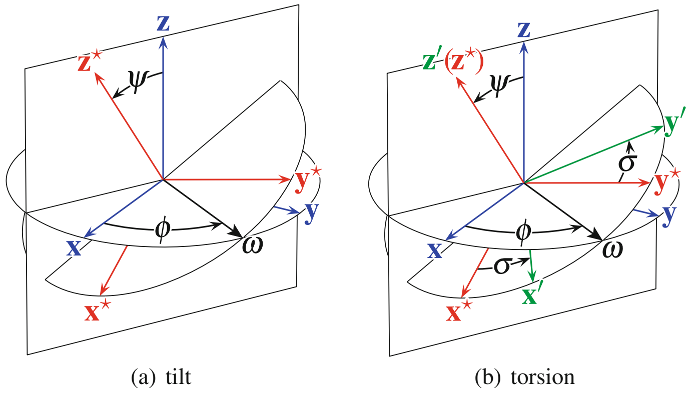


Fig. 1 Tilt and torsion parameterization of SO(3). **a** Tilt: rotation of ψ about axis ω; **b** torsion: rotation of σ about axis z' (modified from Bonev's original illustration [3], courtesy of Ilian Bonev)

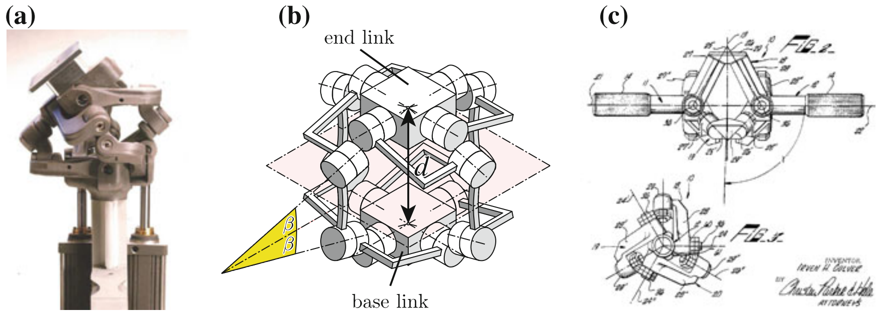


Fig. 2 **a** Mark Rosheim's Omni-wrist III [10] (courtesy of Mark Rosheim). **b** Kinematic scheme of the Omni-wrist III, with four double-universal-joint chains. **c** The Culver joint [11] (source: www.uspto.gov)

comprises three $\mathscr{RRR}$ kinematic chains with concurrent axes ($\mathscr{R}$ denotes a revolute joint). It attains a 140° pointing cone with a maximum of ±30° torsion.

Rosheim developed a different family of parallel wrists called the Omni-wrists [1, 10] (see Fig. 2a, b for Omni-wrist III, which comprises four double-universal-joint chains). They are known to be kinematically equivalent to a class of 2-DoF constant-velocity (CV) couplings for intersecting shafts [11, 12] (see Fig. 2c for the Culver joint, an "Omni-wrist" with three or more double-universal-joint chains). The yaw and pitch axes cannot be defined for such wrists, since their motion pattern [13] is not the same as that of a universal joint. Conversely, it is observed to have a constantly zero torsion-angle parameter [3, 14]. The instantaneous kinematics of CV couplings was systematically studied by Hunt in [12], and more recently in [15]. Hunt observed that the joint screws of a CV kinematic chain must be mirror symmetric about the plane Π which perpendicularly bisects the angle formed by the input and output shaft axes (see Fig. 3a, b for typical 3-DoF CV chains). The instantaneous twist space of

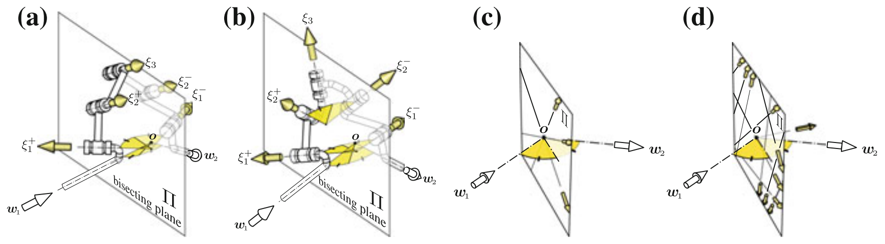


Fig. 3 **a**, **b** Typical CV kinematic chains of a plunging coupling. **c**, **d** Screw systems of a CV coupling with intersecting shafts, where $\boldsymbol{\omega}_1$ and $\boldsymbol{\omega}_2$ are unit vectors along the input and the output shaft axes, respectively: **c** non-plunging coupling, **d** plunging coupling

a CV coupling comprises, at a general configuration, either a pencil of zero-pitch twists converging in one point of Π (non-plunging type, see Fig. 3c) or a field of zero-pitch twists in Π (plunging type, see Fig. 3d). These special twist systems (i.e. subspaces of $\mathfrak{se}(3)$, the Lie algebra of the special Euclidean group SE(3)) are studied in detail in [16]. Carricato considered applications of CV couplings in decoupled and homokinetic closed-chain robotic wrists [15].

Despite having a large singularity-free orientation range, all Omni-wrists display an undesirable parasitic motion [14]. As shown in Fig. 2b, as long as the distance d between the centers of the two universal joints in each chain is nonzero, the motion pattern of the Omni-wrist consists, at any configuration, of a 2-DoF instantaneous rotation about a point whose location is not fixed [15]. Besides, since the links of the Omni-wrists do not undergo pure rotations, mechanical design for link-interference avoidance (see [8]) is more difficult. It is worth emphasizing that all Omni-wrist designs have only tilt (yaw and pitch) capability; an additional roll axis can be installed either at the base or at the end-effector to generate a third torsional DoF.

In this paper, we show that it is possible to eliminate the parasitic motion of the aforementioned CV-coupling-equivalent 2-DoF parallel wrists through a closer investigation of the motion pattern of CV couplings and their associated twist systems. We propose a novel 2-DoF parallel kinematics wrist as a result of this investigation. It is observed that the motion pattern of a CV coupling is exactly the exponential image $\exp(\mathfrak{m})$ of its screw system $\mathfrak{m}$. A recent investigation reveals that $\exp(\mathfrak{m})$ belongs to one of seven classes of symmetric subspaces of SE(3) [17], namely submanifolds M of SE(3) containing the identity and closed under inversion symmetry [18, 19]:

$$\mathbf{g}\mathbf{h}^{-1}\mathbf{g} \in \mathrm{M} \qquad \forall \mathbf{g}, \mathbf{h} \in \mathrm{M} \tag{1}$$

The corresponding twist systems are Lie triple subsystems (LTS) of $\mathfrak{se}(3)$ [20, 21], namely linear subspaces $\mathfrak{m}$ of $\mathfrak{se}(3)$ that are closed under double Lie brackets (or triple product):

$$[\boldsymbol{\xi}, \boldsymbol{\zeta}, \boldsymbol{\varsigma}] \triangleq [[\boldsymbol{\xi}, \boldsymbol{\zeta}], \boldsymbol{\varsigma}] \in \mathfrak{m} \qquad \forall \boldsymbol{\xi}, \boldsymbol{\zeta}, \boldsymbol{\varsigma} \in \mathfrak{m} \tag{2}$$

The mirror symmetry of the joint screws of a CV-coupling-equivalent parallel wrist is shown to be a direct consequence of the involution symmetry of the corresponding symmetric subspaces [18].

Recently, a systematic type synthesis methodology for symmetric subspaces of SE(3) has been developed and the results are being prepared for publication at the moment. Its full description requires the Lie theoretic tools presented in [18], which will not be elaborated again in this paper. Instead, we give a brief review of the main results, and apply them to the synthesis of a 2-DoF parallel wrist family with a large singularity-free orientation range. The geometry of the novel parallel wrist is described, and a screw-theory-based analysis is performed for a less systematic, yet intuitive, understanding of the 2-DoF symmetric subspace generator. An additional torsional DoF may be added to the base or the end-effector of the 2-DoF parallel wrist, thus realizing the full SO(3) motion with a 180° pointing cone and unlimited rolling (torsion). Much freedom for further design modifications is retained, as long as the basic geometry of the underlying symmetric subspace is not violated.

2 A Brief Review of Symmetric Subspace Theory

SE(3) is a symmetric space under the inversion symmetry $S_{\mathbf{g}}(\mathbf{h}) = \mathbf{g}\mathbf{h}^{-1}\mathbf{g}$, $\forall \mathbf{g}, \mathbf{h} \in$ SE(3) [20]. Its (connected) symmetric subspaces, i.e. subsets that are closed under inversion symmetry and contain the identity, are given by the exponentials of LTSs, which are vector subspaces of $\mathfrak{se}(3)$ that are closed under the triple product [18, 19, 21]. There are seven conjugacy classes of (connected) symmetric subspaces of SE(3) (excluding Lie subgroups as trivial symmetric subspaces) [18, 19]. In particular, the 2-dimensional subspace $\langle \widehat{\mathbf{x}}, \widehat{\mathbf{y}} \rangle$[1] of $\mathfrak{so}(3)$, the Lie algebra of SO(3), is a LTS because:

$$[[\widehat{\mathbf{x}}, \widehat{\mathbf{y}}], \widehat{\mathbf{x}}] = [\widehat{\mathbf{z}}, \widehat{\mathbf{x}}] = \widehat{\mathbf{y}} \in \langle \widehat{\mathbf{x}}, \widehat{\mathbf{y}} \rangle \ , \qquad [[\widehat{\mathbf{x}}, \widehat{\mathbf{y}}], \widehat{\mathbf{y}}] = [\widehat{\mathbf{z}}, \widehat{\mathbf{y}}] = -\widehat{\mathbf{x}} \in \langle \widehat{\mathbf{x}}, \widehat{\mathbf{y}} \rangle \tag{3}$$

Hence, $\mathrm{M} = \exp\langle \widehat{\mathbf{x}}, \widehat{\mathbf{y}} \rangle$ is a symmetric subspace of SO(3), and therefore of SE(3). $\exp\langle \widehat{\mathbf{x}}, \widehat{\mathbf{y}} \rangle$ is exactly the motion pattern of the human-eye movement and of non-plunging CV couplings. Several theoretical properties of symmetric subspaces and their corresponding LTSs have been systematically developed in [18]. Here, we give a brief summary, without proof, for the special case of $\mathrm{M} = \exp\langle \widehat{\mathbf{x}}, \widehat{\mathbf{y}} \rangle$.

(SS 1) $\mathfrak{h}_{\langle \widehat{\mathbf{x}}, \widehat{\mathbf{y}} \rangle} := [\langle \widehat{\mathbf{x}}, \widehat{\mathbf{y}} \rangle, \langle \widehat{\mathbf{x}}, \widehat{\mathbf{y}} \rangle] = \langle \widehat{\mathbf{z}} \rangle$ is a Lie subalgebra of $\mathfrak{so}(3)$.

(SS 2) $\mathfrak{g}_{\langle \widehat{\mathbf{x}}, \widehat{\mathbf{y}} \rangle} := \langle \widehat{\mathbf{x}}, \widehat{\mathbf{y}} \rangle + \mathfrak{h}_{\langle \widehat{\mathbf{x}}, \widehat{\mathbf{y}} \rangle} = \mathfrak{so}(3)$ is the completion algebra of $\langle \widehat{\mathbf{x}}, \widehat{\mathbf{y}} \rangle$.

(SS 3) The (right pull-back) tangent space of $\exp\langle \widehat{\mathbf{x}}, \widehat{\mathbf{y}} \rangle$ at $e^{\widehat{\boldsymbol{\omega}}}$, $\widehat{\boldsymbol{\omega}} \in \langle \widehat{\mathbf{x}}, \widehat{\mathbf{y}} \rangle$, obeys the half-angle property:

[1]Here, the wedge operator $\wedge$ takes a 3-dimensional vector $\boldsymbol{\omega} \in \mathbb{R}^3$ into the corresponding skew-symmetric matrix $\widehat{\boldsymbol{\omega}}$, so that $\widehat{\boldsymbol{\omega}}\mathbf{v} = \boldsymbol{\omega} \times \mathbf{v}$, $\forall \boldsymbol{\omega}, \mathbf{v} \in \mathbb{R}^3$. In this paper, we use $\mathbf{x}$, $\mathbf{y}$ and $\mathbf{z}$ to denote the unit coordinate axis vectors $(1, 0, 0)^T$, $(0, 1, 0)^T$ and $(0, 0, 1)^T$ of $\mathbb{R}^3$. We also use the angled bracket $\langle , \rangle$ to denote a vector subspace spanned by a set of vectors.

$$T_{e^{\widehat{\boldsymbol{\omega}}}} \exp\langle \widehat{\mathbf{x}}, \widehat{\mathbf{y}} \rangle = \mathrm{Ad}_{e^{\widehat{\boldsymbol{\omega}}/2}} \langle \widehat{\mathbf{x}}, \widehat{\mathbf{y}} \rangle = \langle \widehat{e^{\widehat{\boldsymbol{\omega}}/2}\mathbf{x}}, \widehat{e^{\widehat{\boldsymbol{\omega}}/2}\mathbf{y}} \rangle \ , \quad \forall \widehat{\boldsymbol{\omega}} \in \langle \widehat{\mathbf{x}}, \widehat{\mathbf{y}} \rangle \tag{4}$$

where $\mathrm{Ad}_{\mathbf{R}}(\cdot)$, $\mathbf{R} \in SO(3)$ is the Adjoint transformation on $\mathfrak{so}(3)$ defined by [22]

$$\mathrm{Ad}_{\mathbf{R}} \widehat{\boldsymbol{\omega}} = \mathbf{R} \widehat{\boldsymbol{\omega}} \mathbf{R}^T \tag{5}$$

(SS 4) $\langle \widehat{\mathbf{x}}, \widehat{\mathbf{y}} \rangle$ is invariant under the Adjoint action of the subgroup $\exp\langle \widehat{\mathbf{z}} \rangle$:

$$\mathrm{Ad}_{e^{\lambda \widehat{\mathbf{z}}}} \langle \widehat{\mathbf{x}}, \widehat{\mathbf{y}} \rangle = \langle \widehat{\mathbf{x}}, \widehat{\mathbf{y}} \rangle \ , \qquad \forall \lambda \in \mathbb{R} \tag{6}$$

and similarly $\exp\langle \widehat{\mathbf{x}}, \widehat{\mathbf{y}} \rangle$ is invariant under conjugation by elements of the subgroup $\exp\langle \widehat{\mathbf{z}} \rangle$:

$$e^{\lambda \widehat{\mathbf{z}}} \exp\langle \widehat{\mathbf{x}}, \widehat{\mathbf{y}} \rangle e^{-\lambda \widehat{\mathbf{z}}} = \exp\langle \widehat{\mathbf{x}}, \widehat{\mathbf{y}} \rangle \ , \quad \forall \lambda \in \mathbb{R} \tag{7}$$

(SS 5) $\mathfrak{g}_{\langle \widehat{\mathbf{x}}, \widehat{\mathbf{y}} \rangle} = \langle \widehat{\mathbf{x}}, \widehat{\mathbf{y}} \rangle + \mathfrak{h}_{\langle \widehat{\mathbf{x}}, \widehat{\mathbf{y}} \rangle}$ provides a parameterization of the completion group $\exp \mathfrak{g}_{\langle \widehat{\mathbf{x}}, \widehat{\mathbf{y}} \rangle} = SO(3)$, namely

$$\begin{aligned} \langle \widehat{\mathbf{x}}, \widehat{\mathbf{y}} \rangle \times \mathfrak{h}_{\langle \widehat{\mathbf{x}}, \widehat{\mathbf{y}} \rangle} &\to \exp \mathfrak{g}_{\langle \widehat{\mathbf{x}}, \widehat{\mathbf{y}} \rangle} \ , \\ (\widehat{\boldsymbol{\omega}}, \lambda \widehat{\mathbf{z}}) &\mapsto e^{\widehat{\boldsymbol{\omega}}} e^{\lambda \widehat{\mathbf{z}}} \end{aligned} \tag{8}$$

(SS 6) The symmetric subspace $\exp\langle \widehat{\mathbf{x}}, \widehat{\mathbf{y}} \rangle$ can be generated by the *symmetric movement* of a *symmetric chain*:

$$\begin{aligned} e^{\theta_1 \widehat{\boldsymbol{\omega}}_1^+} e^{\theta_2 \widehat{\boldsymbol{\omega}}_2^+} \cdot e^{\theta_2 \widehat{\boldsymbol{\omega}}_2^-} e^{\theta_1 \widehat{\boldsymbol{\omega}}_1^-} &= e^{\theta_1 \widehat{\boldsymbol{\omega}}_1^+} e^{\theta_2 \widehat{\boldsymbol{\omega}}_2^+} (e^{-\theta_1 \widehat{\boldsymbol{\omega}}_1^-} e^{-\theta_2 \widehat{\boldsymbol{\omega}}_2^-})^{-1} \\ &= e^{\widehat{\boldsymbol{\omega}}} e^{\lambda \widehat{\mathbf{z}}} (e^{-\widehat{\boldsymbol{\omega}}} e^{\lambda \widehat{\mathbf{z}}})^{-1} = e^{2\widehat{\boldsymbol{\omega}}} \end{aligned} \tag{9}$$

for some $\boldsymbol{\omega} \in \langle \mathbf{x}, \mathbf{y} \rangle$, $\lambda \in \mathbb{R}$, where each *symmetric pair* of joint axes $(\boldsymbol{\omega}_i^+, \boldsymbol{\omega}_i^-)$, $i = 1, 2$ is given by:

$$\begin{cases} \boldsymbol{\omega}_i^+ = e^{\widehat{\mathbf{u}}_i} \boldsymbol{\omega}_i, \\ \boldsymbol{\omega}_i^- = e^{-\widehat{\mathbf{u}}_i} \boldsymbol{\omega}_i \end{cases} \qquad \widehat{\boldsymbol{\omega}}_i, \widehat{\mathbf{u}}_i \in \langle \widehat{\mathbf{x}}, \widehat{\mathbf{y}} \rangle \qquad \text{(geometric condition)} \tag{10}$$

In other words, $\boldsymbol{\omega}_i^+$ and $\boldsymbol{\omega}_i^-$ are generated by rotating $\boldsymbol{\omega}_i$ about another axis $\mathbf{u}_i \in \langle \mathbf{x}, \mathbf{y} \rangle$ with magnitude $\|\mathbf{u}_i\|$ and $-\|\mathbf{u}_i\|$, respectively, thereby forming a mirror symmetry about the XY plane. Alternatively,

$$\begin{cases} \boldsymbol{\omega}_i^+ = \boldsymbol{\omega}_i + \lambda \mathbf{z}, \\ \boldsymbol{\omega}_i^- = \boldsymbol{\omega}_i - \lambda \mathbf{z} \end{cases} \qquad \widehat{\boldsymbol{\omega}}_i \in \langle \widehat{\mathbf{x}}, \widehat{\mathbf{y}} \rangle, \lambda \in \mathbb{R} \qquad \text{(algebraic condition)} \tag{11}$$

with

$$\begin{gathered}\langle \widehat{\boldsymbol{\omega}}_1^+, \widehat{\boldsymbol{\omega}}_2^+ \rangle \oplus \langle \widehat{\mathbf{z}} \rangle = \mathfrak{g}_{\langle \widehat{\mathbf{x}}, \widehat{\mathbf{y}} \rangle} \quad \text{or} \\ \langle \widehat{\boldsymbol{\omega}}_1^-, \widehat{\boldsymbol{\omega}}_2^- \rangle \oplus \langle \widehat{\mathbf{z}} \rangle = \mathfrak{g}_{\langle \widehat{\mathbf{x}}, \widehat{\mathbf{y}} \rangle} \quad \text{or} \\ \langle \widehat{\boldsymbol{\omega}}_1, \widehat{\boldsymbol{\omega}}_2 \rangle = \langle \widehat{\mathbf{x}}, \widehat{\mathbf{y}} \rangle \end{gathered} \tag{12}$$

(SS 7) The symmetric movement of a symmetric chain satisfies the *half-angle property*, namely

$$\begin{gathered}\mathrm{Ad}_{e^{\theta_1 \widehat{\boldsymbol{\omega}}_1^+} e^{\theta_2 \widehat{\boldsymbol{\omega}}_2^+} e^{\theta_2 \widehat{\boldsymbol{\omega}}_2^-} e^{\theta_1 \widehat{\boldsymbol{\omega}}_1^-}} \mathbf{z} = \mathrm{Ad}_{e^{2\widehat{\boldsymbol{\omega}}}} \mathbf{z} = e^{2\widehat{\boldsymbol{\omega}}} \mathbf{z} \\ \mathrm{Ad}_{e^{\theta_1 \widehat{\boldsymbol{\omega}}_1^+} e^{\theta_2 \widehat{\boldsymbol{\omega}}_2^+}} \mathbf{z} = \mathrm{Ad}_{e^{\widehat{\boldsymbol{\omega}}}} \mathbf{z} = e^{\widehat{\boldsymbol{\omega}}} \mathbf{z} \end{gathered} \tag{13}$$

3 A Novel 2-DoF Parallel Wrist

The wrist presented in this Section has a fully spherical architecture, with all hinge axes converging in a common point. Accordingly, in order to ease notation, all axes will be simply denoted by unit vectors parallel to them. As shown in Fig. 4a, b at the initial configuration, the wrist consists of three (or more) $\mathscr{R}\mathscr{R}\mathscr{R}$ chains, with the first and third joint axes $\boldsymbol{\omega}_{i1}$ and $\boldsymbol{\omega}_{i3}$ of each chain i being mirror symmetric about a common bisecting plane Π, and $\boldsymbol{\omega}_{i2}$ lying in Π. We denote the joint variable about axis $\boldsymbol{\omega}_{ij}$ by θ_{ij}. Between the proximal link (connected to the base link via joint $\boldsymbol{\omega}_{i1}$, $i = 1, 2, 3$) and the distal link (connected to the end link via joint $\boldsymbol{\omega}_{i3}$, $i = 1, 2, 3$) of each chain, an extra $\mathscr{R}\mathscr{R}$ interconnecting link connects the pivot of joint $\boldsymbol{\omega}_{i2}$ to a central shaft having its axis $\boldsymbol{\omega}_c$ perpendicular to Π. The plane of symmetry Π is highlighted by a pink dashed circle in Fig. 4a. If no interconnecting links are imposed, the mechanism is a 3-DoF SO(3) generator (as shown in Fig. 4b). The interconnection is separately shown in Fig. 4c. According to the mirror symmetry, the proximal and distal link in each chain are identical. For a symmetrical kinematic behavior, the $\mathscr{R}\mathscr{R}\mathscr{R}$ chains share the same geometry, with each one being obtained from the previous one, at the home configuration, by a 120° rotation about $\boldsymbol{\omega}_c$. Therefore, the geometry of the wrist is completely determined by two parameters, namely:

(1) the angle α between the two revolute joint axes $\boldsymbol{\omega}_{i1}, \boldsymbol{\omega}_{i2}$ (or $\boldsymbol{\omega}_{i3}, \boldsymbol{\omega}_{i2}$) of the proximal link (or the distal link);
(2) the angle β, $\beta \neq 0$, formed by Π and the plane spanned by $\boldsymbol{\omega}_{i1}, \boldsymbol{\omega}_{i2}$ (or $\boldsymbol{\omega}_{i3}, \boldsymbol{\omega}_{i2}$) at the home configuration.

Without loss of generality, we define the two actuation joints to be those corresponding to axes $\boldsymbol{\omega}_{11}$ and $\boldsymbol{\omega}_{21}$. We set up the reference frame at the initial configuration[2] with the $\mathbf{x}$-axis coinciding with $\boldsymbol{\omega}_{12}^0$, and the $\mathbf{z}$-axis coinciding with $\boldsymbol{\omega}_c^0$. It is not

[2]We use a superscript ^{0} to distinguish a joint axis at the initial configuration from its value at a generic configuration.

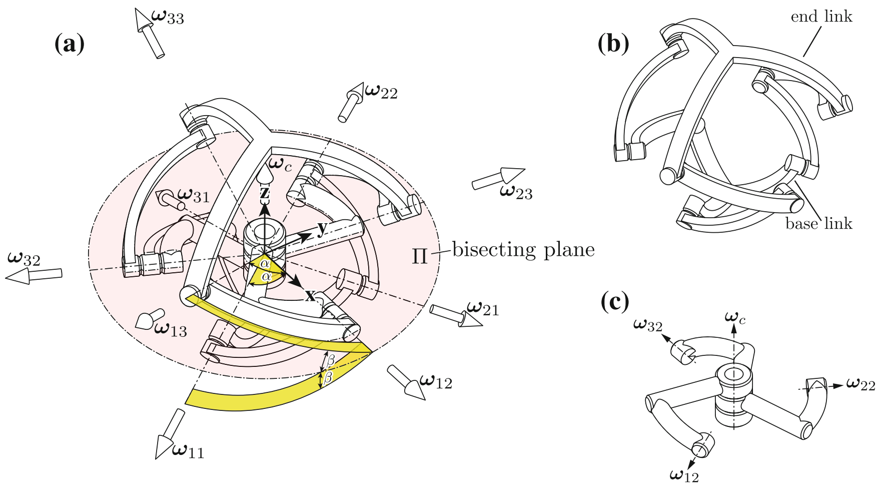


Fig. 4 Geometry and notation of the novel 2-DoF parallel wrist. **a** The joint screws of all chains are mirror-symmetric about a common bisecting plane Π; **b** the wrist without interconnecting links has a 3-$\mathscr{RRR}$ architecture; **c** the interconnecting links are connected via revolute joints to a common shaft with axis $\boldsymbol{\omega}_c$ perpendicular to Π

difficult to see that the initial joint axes are given by:

$$\begin{aligned}
\boldsymbol{\omega}_{11} &= e^{\beta\widehat{\mathbf{x}}}e^{-\alpha\widehat{\mathbf{z}}}\mathbf{x}\ , & \boldsymbol{\omega}_{21} &= e^{2\pi/3\widehat{\mathbf{z}}}\boldsymbol{\omega}_{11}\ , & \boldsymbol{\omega}_{31} &= e^{4\pi/3\widehat{\mathbf{z}}}\boldsymbol{\omega}_{11}\ , \\
\boldsymbol{\omega}^0_{12} &= \mathbf{x}\ , & \boldsymbol{\omega}^0_{22} &= e^{2\pi/3\widehat{\mathbf{z}}}\boldsymbol{\omega}^0_{12}\ , & \boldsymbol{\omega}^0_{32} &= e^{4\pi/3\widehat{\mathbf{z}}}\boldsymbol{\omega}^0_{12}\ , \\
\boldsymbol{\omega}^0_{13} &= e^{-\beta\widehat{\mathbf{x}}}e^{-\alpha\widehat{\mathbf{z}}}\mathbf{x}\ , & \boldsymbol{\omega}^0_{23} &= e^{2\pi/3\widehat{\mathbf{z}}}\boldsymbol{\omega}^0_{13}\ , & \boldsymbol{\omega}^0_{33} &= e^{4\pi/3\widehat{\mathbf{z}}}\boldsymbol{\omega}^0_{13}\ ,
\end{aligned} \tag{14}$$

where $\boldsymbol{\omega}_{i1}$ and $\boldsymbol{\omega}_{i3}$ respect the geometric condition (10), and $\boldsymbol{\omega}_{i2}$ is located in the bisecting plane Π, thus being self-symmetric.

The motion pattern of the wrist shown in Fig. 4 is the 2-dimensional symmetric subspace $\exp\langle\widehat{\mathbf{x}}, \widehat{\mathbf{y}}\rangle$:

(1) according to (9) in **(SS 6)**, it generates $\exp\langle\widehat{\mathbf{x}}, \widehat{\mathbf{y}}\rangle$, as long as all three chains undergo symmetric movement, i.e. $\theta_{i1} = \theta_{i3}$, $i = 1, 2, 3$;
(2) according to (13) in **(SS 7)**, for any end-effector displacement $e^{2\boldsymbol{\omega}}$, $\boldsymbol{\omega} \in \langle\mathbf{x}, \mathbf{y}\rangle$, the displacement of the central shaft (with axis $\boldsymbol{\omega}_c$) is always $e^{\boldsymbol{\omega}}$ when it follows any of the subchains $(\boldsymbol{\omega}_{11}, \boldsymbol{\omega}_{12})$, $(\boldsymbol{\omega}_{21}, \boldsymbol{\omega}_{22})$ and $(\boldsymbol{\omega}_{31}, \boldsymbol{\omega}_{32})$; therefore, the interconnecting links do not interfere with the symmetric movement;
(3) it is possible to verify that, at the initial configuration, the wrist is constrained to have an instantaneous mobility equal to 2, and this is necessarily full-cycle (see for example the proof of [23]-Proposition 6), and represented by the symmetric movement in (1).

The idea of adding interconnecting links comes from recent innovative designs of CV shaft couplings [24, 25], for which we claim no originality. Our contribution lies in a (non-instantaneous) validation of its full-cycle mobility using the half-angle property in **(SS 7)**. The generality of properties **(SS 1)–(SS 7)** [18] also implies that such a design procedure can be generalized to other symmetric subspaces.

4 Kinematic Analysis of the Novel Parallel Wrist

The kinematic analysis of the wrist described in Sect. 3, equipped with interconnecting chains, is more complex than that of a simple 3-$\mathscr{RRR}$ spherical mechanism. However, we can use the half-angle property and other properties introduced in Sect. 2 to simplify the analysis.

4.1 Inverse and Direct Kinematics

The inverse and direct kinematics may be solved in closed form. Given the end link orientation $\mathbf{R} = e^{2\psi\widehat{\boldsymbol{\omega}}}$, with $\boldsymbol{\omega} = \mathbf{x}\cos\phi + \mathbf{y}\cos\phi$, according to the algebraic condition (11), $\boldsymbol{\omega}_{i1} + \boldsymbol{\omega}_{i3}$ and $\boldsymbol{\omega}_{i2}$ are always included in the bisecting plane Π, which tilts half as much as the end link, and $\boldsymbol{\omega}_c$ is always perpendicular to Π. Therefore, from

$$(\boldsymbol{\omega}_{i1} + \boldsymbol{\omega}_{i3})^T \boldsymbol{\omega}_{i2} = 2\cos\alpha \tag{15}$$

we have:

$$\boldsymbol{\omega}_c = e^{\psi\widehat{\boldsymbol{\omega}}}\mathbf{z}\ , \qquad \theta = \cos^{-1}\left(\frac{2\cos\alpha}{\|\boldsymbol{\omega}_{i1} + \boldsymbol{\omega}_{i3}\|}\right)\ , \qquad \boldsymbol{\omega}_{i2} = e^{\theta\widehat{\boldsymbol{\omega}}_c}\frac{\boldsymbol{\omega}_{i1} + \boldsymbol{\omega}_{i3}}{\|\boldsymbol{\omega}_{i1} + \boldsymbol{\omega}_{i3}\|} \tag{16}$$

where $\boldsymbol{\omega}_{i1}$ and $\boldsymbol{\omega}_{i3}$ are known. The unknown joint input variables θ_{11} and θ_{21} can be easily derived by solving

$$\boldsymbol{\omega}_{i2} = e^{\theta_{i1}\widehat{\boldsymbol{\omega}}_{i1}}\boldsymbol{\omega}_{i2}^0\ ,\ \ i = 1, 2 \tag{17}$$

When solving the direct kinematics, instead, the actuator variables θ_{11} and θ_{21} are given, so that $\boldsymbol{\omega}_{12}$ and $\boldsymbol{\omega}_{22}$ can be immediately computed by (17). Again, using the fact that $\boldsymbol{\omega}_c$ is perpendicular to $\boldsymbol{\omega}_{i2},\ \ i = 1, 2, 3$, we have

$$\boldsymbol{\omega}_c = \frac{\boldsymbol{\omega}_{12} \times \boldsymbol{\omega}_{22}}{\|\boldsymbol{\omega}_{12} \times \boldsymbol{\omega}_{22}\|} = e^{\psi\widehat{\boldsymbol{\omega}}}\mathbf{z} \tag{18}$$

The end link orientation is finally calculated as $\mathbf{R} = e^{2\psi\widehat{\boldsymbol{\omega}}}$, according to the half-angle property.

4.2 Singularity Analysis

The full-cycle mobility of the wrist is 2, which is characterized by the symmetric movement of all $\mathscr{RRR}$ chains. The wrist may leave the symmetric movement at either a *passive-constraint singularity* or a *leg singularity* [5]. When symmetric movement is not violated, singularity analysis can be separately performed on the proximal or distal half of the wrist, each forming a parallel mechanism with the interconnecting links.

4.2.1 Singularities of the Proximal Half

As shown in Fig. 5a, the proximal half is a parallel mechanism comprising one $\mathscr{RR}$ chain with joint axes $(\boldsymbol{\omega}_{11}, \boldsymbol{\omega}_{12})$, and two $\mathscr{RRR}$ chains with joint axes $(\boldsymbol{\omega}_{21}, \boldsymbol{\omega}_{22}, \boldsymbol{\omega}_c)$ and $(\boldsymbol{\omega}_{31}, \boldsymbol{\omega}_{32}, \boldsymbol{\omega}_c)$, respectively. The proximal half can only admit a leg singularity in leg 2 or 3. Since $\boldsymbol{\omega}_c$ is always perpendicular to $\boldsymbol{\omega}_{i2}$, $i = 2, 3$, leg singularity occurs when $\boldsymbol{\omega}_{i1}$, $\boldsymbol{\omega}_{i2}$ and $\boldsymbol{\omega}_c$ become coplanar in a plane perpendicular to the bisecting plane Π (see Fig. 8b). This case is covered in Sect. 4.2.3.

A configuration of actuation singularity (or *active-constraint singularity*, [5]) corresponds to the mechanism increasing its mobility after the input joints $\boldsymbol{\omega}_{11}$ and $\boldsymbol{\omega}_{21}$ are locked. This occurs when the three constraint torques $\boldsymbol{\tau}_1$, $\boldsymbol{\tau}_2$ and $\boldsymbol{\tau}_3$ shown in Fig. 5b are linearly dependent. The constraint torques in each chain should be perpendicular to all its joint axes (see for example [16]):

$$\boldsymbol{\omega}_{12}^T\boldsymbol{\tau}_1 = 0\ , \qquad \boldsymbol{\omega}_{12}^T\boldsymbol{\tau}_2 = 0\ , \qquad \boldsymbol{\omega}_{22}^T\boldsymbol{\tau}_3 = 0\ , \qquad \boldsymbol{\omega}_c^T\boldsymbol{\tau}_3 = 0\ . \tag{19}$$

Since, without loss of generality, $\boldsymbol{\tau}_1$ and $\boldsymbol{\tau}_2$ may be chosen, respectively, perpendicular and parallel to $\boldsymbol{\omega}_c$, $\boldsymbol{\tau}_2$ and $\boldsymbol{\tau}_3$ are always perpendicular. It follows that $\boldsymbol{\tau}_1$, $\boldsymbol{\tau}_2$ and $\boldsymbol{\tau}_3$ are linearly dependent when $\boldsymbol{\tau}_1$ and $\boldsymbol{\tau}_3$ are parallel, i.e.

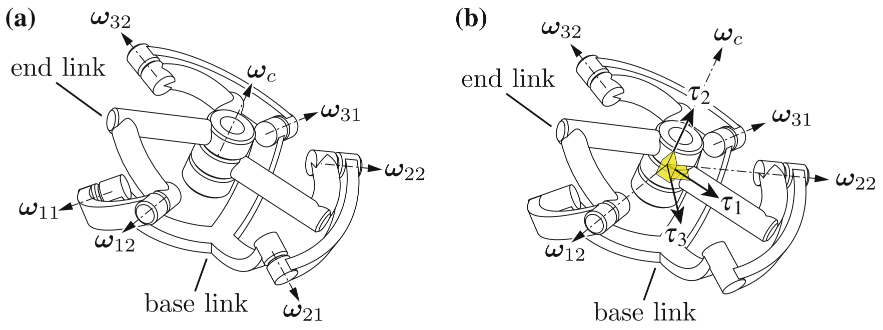


Fig. 5 **a** The proximal half of the novel wrist; **b** constraint torques of the mechanism with input joint variables θ_{11} and θ_{21} locked

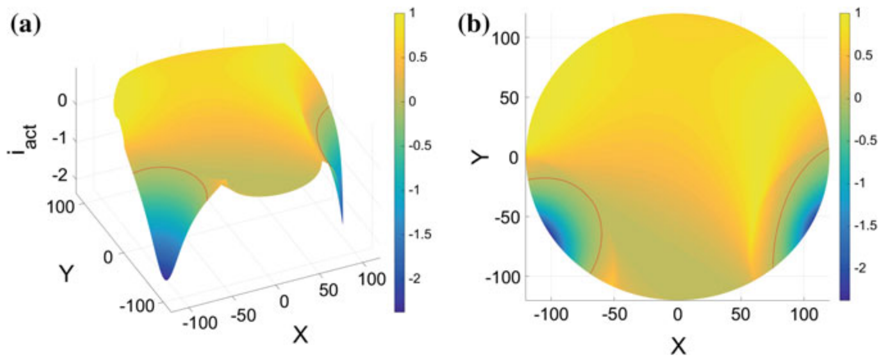


Fig. 6 Plot of actuation singularity index i_{act} against tilt angle $2\psi \in [-120°, 120°]$ ($\alpha = 60°$, $\beta = 25°$). The *red* level curve denotes the actuation singularity loci. i_{act} taking a negative value is due to failure to acquire an inverse kinematics solution

$$\boldsymbol{\omega}_{12} = \pm\boldsymbol{\omega}_{22} \tag{20}$$

An immediate actuation singularity index i_{act} can be defined as follows:

$$i_{\text{act}} = 1 - |\boldsymbol{\omega}_{12}^T \boldsymbol{\omega}_{22}| \tag{21}$$

Figure 6 shows the variation of i_{act} with respect to different tilt axis $\boldsymbol{\omega} = \mathbf{x}\cos\phi + \mathbf{y}\sin\phi$ and angle 2ψ in Cartesian coordinates, i.e. $\mathbf{X} = 2\psi\cos\phi$, $\mathbf{Y} = 2\psi\sin\phi$, for a kinematic geometry of $\alpha = 60°$, $\beta = 25°$.

4.2.2 Singularities of the Distal Half

The distal half of the wrist is shown in Fig. 7. Out of actuation singularities, when the input joint variables are locked, the interconnecting joints with axis $\boldsymbol{\omega}_c$ are not movable. Therefore, the three interconnecting links become a single link, which can be considered as the base link of an equivalent parallel mechanism. This distal mechanism comprises three $\mathscr{R}\mathscr{R}$ chains with joints axes $(\boldsymbol{\omega}_{12}, \boldsymbol{\omega}_{13})$, $(\boldsymbol{\omega}_{22}, \boldsymbol{\omega}_{23})$ and $(\boldsymbol{\omega}_{32}, \boldsymbol{\omega}_{33})$. A configuration of *static singularity* [5] emerges when the three constraint torques $\boldsymbol{\tau}_4, \boldsymbol{\tau}_5, \boldsymbol{\tau}_6$ become linearly dependent, or equivalently:

$$\det(\boldsymbol{\omega}_{12} \times \boldsymbol{\omega}_{13}, \boldsymbol{\omega}_{22} \times \boldsymbol{\omega}_{23}, \boldsymbol{\omega}_{32} \times \boldsymbol{\omega}_{33}) = 0 \tag{22}$$

An instance of static singularity is shown in Fig. 8a. In the figure, the joint screws of chains 1 and 2 collapse into the bisecting plane Π. Consequently, their constraint torques, denoted $\boldsymbol{\tau}_4, \boldsymbol{\tau}_5$, become linearly dependent.

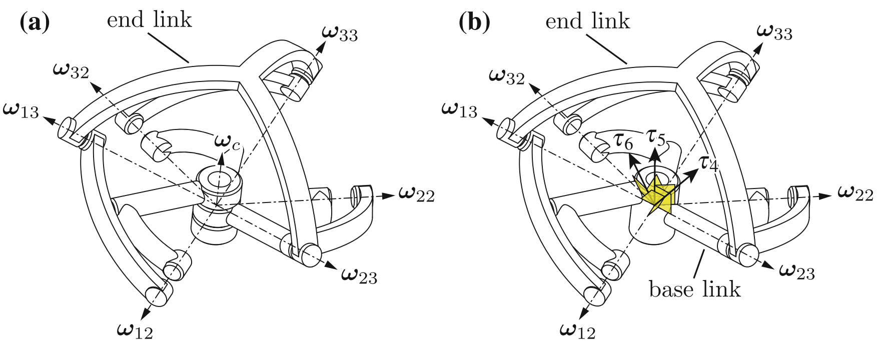


Fig. 7 **a** The distal half of the novel wrist; **b** constraint torques of the mechanism with input joint variables $(\theta_{11}, \theta_{21})$ locked

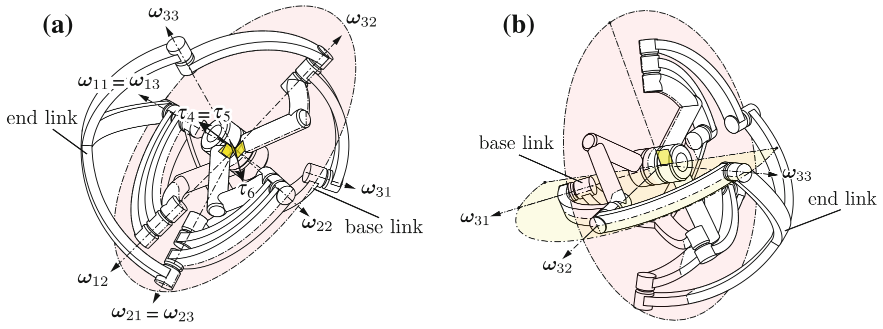


Fig. 8 **a** A static singularity configuration and **b** a stationary configuration of the wrist

4.2.3 Stationary Configurations

Again by the half-angle property, the wrist is at a *stationary configuration* [5] when $\boldsymbol{\omega}_{i1} + \boldsymbol{\omega}_{i3}$ and $\boldsymbol{\omega}_{i2}$ become linearly dependent for one or more chains i. This occurs when $|\theta_{i2}| = 180° - 2\beta$ (see Fig. 8b). Therefore, in order to achieve a singularity-free pointing cone of 180°, $180° - 2\beta$ must be larger than 90°, or $\beta < 45°$.

Stationary singularity also occurs when $\boldsymbol{\omega}_{i1} = \boldsymbol{\omega}_{i3}$ for all $\mathscr{RRR}$ chains. The leg twist spaces degenerate into three planar pencils of zero-pitch screws spanned by $(\boldsymbol{\omega}_{11}, \boldsymbol{\omega}_{12})$, $(\boldsymbol{\omega}_{21}, \boldsymbol{\omega}_{22})$ and $(\boldsymbol{\omega}_{31}, \boldsymbol{\omega}_{32})$, respectively. If the three pencils have trivial intersection (as shown in Fig. 9a), the end-link becomes immobile. Nevertheless, the proximal half of the wrist still admits 2 DoFs. Since $\boldsymbol{\omega}_{i1}$ and $\boldsymbol{\omega}_{i3}$ remain collinear after finite displacement of the interconnecting links for $i = 1, 2, 3$, the stationary singularity is necessarily finite.

It can also be shown that the above case occurs only when $\beta = 0$, where at the initial configuration (as shown in Fig. 9b) all leg twist spaces are identical to the same planar pencil of zero-pitch screws, namely $\langle \mathbf{x}, \mathbf{y} \rangle$. In this configuration, the revolute joints aligned with $\boldsymbol{\omega}_c$ are transitorily inactive, so that the three interconnecting links

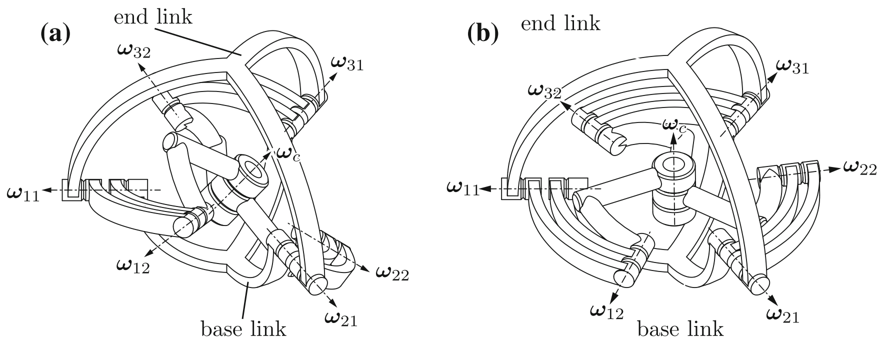


Fig. 9 **a** A stationary configuration of the wrist, with the end-link being motionless and the proximal half having 2 freedoms; **b** the collapsing of proximal and distal links at initial configuration with $\beta = 0$

have no relative motion and can be instantaneously considered as a single link. The twist space of the latter, being the intersection of three copies of $\langle \mathbf{x}, \mathbf{y} \rangle$, is equal to $\langle \mathbf{x}, \mathbf{y} \rangle$. Similarly, the end-link twist space is also equal to $\langle \mathbf{x}, \mathbf{y} \rangle$. The wrist has an instantaneous mobility of 4, gaining an *increased instantaneous mobility* [5] of order 2.

4.3 Jacobian of the 3-DoF Serial-Parallel Wrist (With an Additional Roll Axis)

An extra rolling axis can be added to the base link of the parallel wrist to provide a complete 3-DoF rotation capability. Aside from the aforesaid singularities, the serial-parallel wrist may experience an additional stationary singularity when the roll axis $\boldsymbol{\omega}_r$ (see Fig. 10) is linearly dependent with the bisecting plane Π of the parallel wrist. However, according to the half-angle property, Π tilts only up to 45° within the 180° pointing cone of the parallel wrist. Therefore, $\boldsymbol{\omega}_r$ can never be linearly dependent with Π, and the stationary singularity of the 3-DoF wrist as a whole is eliminated.

By applying both the half-angle property **(SS 7)** and the kinetostatic analysis method introduced in [26], the Jacobian analysis of the 3-DoF parallel wrist can be carried out as follows.

- The constraint torque $\boldsymbol{\tau}_c$ is parallel to $\boldsymbol{\omega}_c$, which is always perpendicular to the bisecting plane Π;
- The actuation torques of $\boldsymbol{\omega}_{11}$ and $\boldsymbol{\omega}_{21}$ are given by:

$$\boldsymbol{\tau}_{a1} = \boldsymbol{\omega}_{12} \times \boldsymbol{\omega}_c \ , \qquad \boldsymbol{\tau}_{a2} = \boldsymbol{\omega}_{22} \times \boldsymbol{\omega}_c . \tag{23}$$

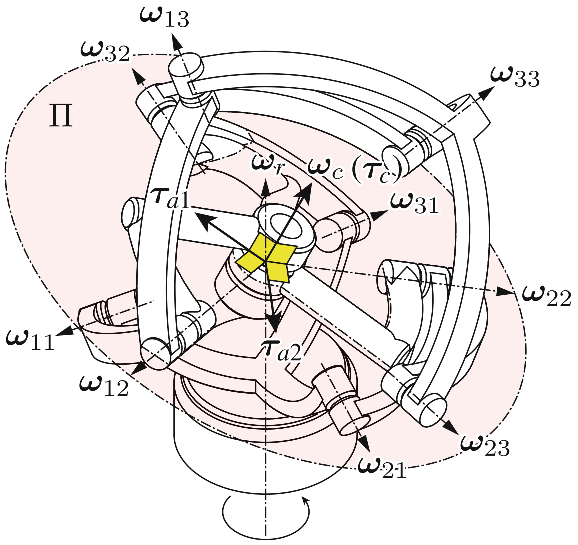


Fig. 10 A serial-parallel 3-DoF wrist with constraint torque $\boldsymbol{\tau}_c$ always perpendicular to Π and actuation torques $\boldsymbol{\tau}_{a1}$, $\boldsymbol{\tau}_{a2}$ lying in Π begin perpendicular to $\boldsymbol{\omega}_{12}$ and $\boldsymbol{\omega}_{22}$, respectively

- The generalized velocity equation of the 2-DoF parallel wrist is given by (see Fig. 10):

$$\underbrace{\begin{pmatrix} \boldsymbol{\tau}_{a1}^T \\ \boldsymbol{\tau}_{a2}^T \\ \boldsymbol{\tau}_{c}^T \end{pmatrix}}_{\mathbf{J}_{dir}} \boldsymbol{\omega}_e = \underbrace{\begin{pmatrix} \boldsymbol{\tau}_{a1}^T(\boldsymbol{\omega}_{11}+\boldsymbol{\omega}_{13}) & 0 \\ 0 & \boldsymbol{\tau}_{a2}^T(\boldsymbol{\omega}_{21}+\boldsymbol{\omega}_{23}) \\ 0 & 0 \end{pmatrix}}_{\mathbf{J}_{inv}} \begin{pmatrix} \dot{\theta}_{11} \\ \dot{\theta}_{21} \end{pmatrix} \tag{24}$$

  where $\boldsymbol{\omega}_e$ is the instantaneous velocity of the end link. Note that the direct kinematics Jacobian $\mathbf{J}_{dir}$ is degenerate when $\boldsymbol{\omega}_{12} \times \boldsymbol{\omega}_c$, $\boldsymbol{\omega}_{22} \times \boldsymbol{\omega}_c$ and $\boldsymbol{\omega}_c$ are linearly dependent, in particular when either $\boldsymbol{\omega}_{12} = \pm\boldsymbol{\omega}_{22}$ (see Sect. 4.2.1) or $\boldsymbol{\tau}_c = \mathbf{0}$. Also, the inverse kinematics Jacobian $\mathbf{J}_{inv}$ is singular, and thus a stationary configuration occurs, when $\boldsymbol{\tau}_{ai}$ is perpendicular to $(\boldsymbol{\omega}_{i1} + \boldsymbol{\omega}_{i3})$, i.e. when $(\boldsymbol{\omega}_{i1} + \boldsymbol{\omega}_{i3})$ and $\boldsymbol{\omega}_{i2}$ are linearly dependent, as shown in Sect. 4.2.3.
- Away from a static singularity, the end link velocity $\tilde{\boldsymbol{\omega}}_e$ of the 3-DoF serial-parallel wrist is given by:

$$\tilde{\boldsymbol{\omega}}_e = \mathbf{z}\dot{\theta}_r + e^{\theta_r\widehat{\mathbf{z}}}\boldsymbol{\omega}_e = \mathbf{z}\dot{\theta}_r + e^{\theta_r\widehat{\mathbf{z}}}\mathbf{J}_o^{-1}\mathbf{J}_\theta \begin{pmatrix} \dot{\theta}_{11} \\ \dot{\theta}_{21} \end{pmatrix} = \left(\mathbf{z}, e^{\theta_r\widehat{\mathbf{z}}}\mathbf{J}_o^{-1}\mathbf{J}_\theta\right) \begin{pmatrix} \dot{\theta}_r \\ \dot{\theta}_{11} \\ \dot{\theta}_{21} \end{pmatrix} \tag{25}$$

5 Tentative Designs with Gear Transmission

Unlike Omni-wrists that have a parasitic motion, the links of the novel wrist undergo pure rotations and move within spherical shells. Therefore, design for link-interference avoidance [8] becomes significantly easier. So long as two physical links are located on different spherical shells, no interference will occur for any configuration. Two tentative designs with bevel gear transmission, one with three and the other with four chains, are is shown in Fig. 11. Both designs locate the roll axis at the base of the 2-DoF parallel wrist, in order to simplify its actuation, thus forming a 3-DoF serial-parallel spherical robotic wrist. Nevertheless, it is possible to locate the roll axis on the end-link of the 2-DoF wrist. In this case, the roll axis may be actuated by a constant-velocity transmission (see for example [15]) passing through the center of the wrist.

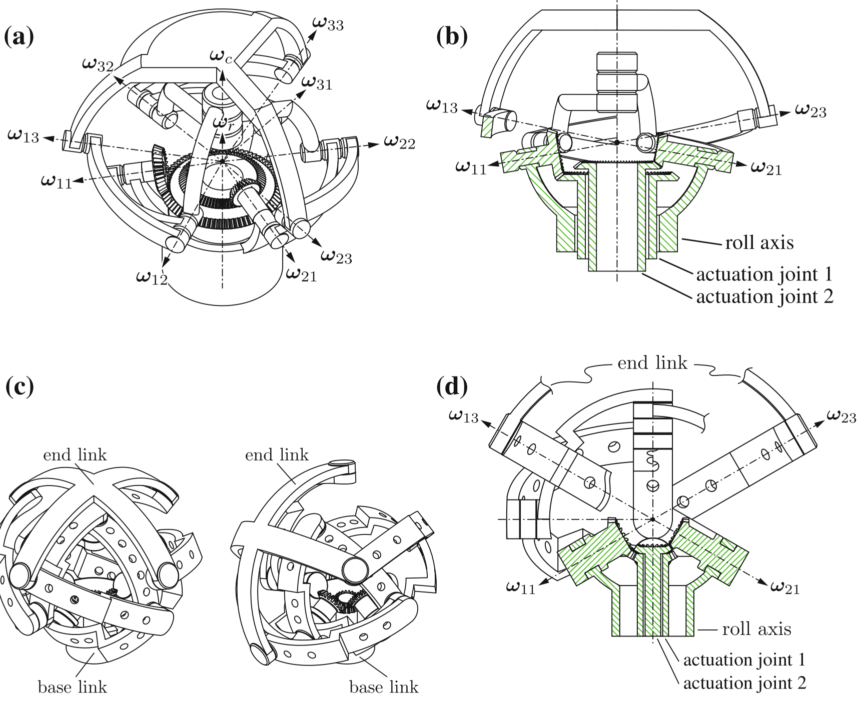


Fig. 11 3-DoF serial-parallel wrist with bevel gear transmission: **a**, **b** three chains; **c**, **d** four chains

6 Conclusions

In this paper, we presented a novel class of 2-DoF parallel wrists based on the 2-dimensional symmetric subspace $\exp\langle\widehat{\mathbf{x}}, \widehat{\mathbf{y}}\rangle$. In comparison to 3-DoF purely parallel wrists and 2-DoF double-universal-joint type parallel wrists, the novel wrist is purely rotational and can be designed in a more compact way. The existence of interconnecting links reduces the occurrence of constraint singularities, without jeopardizing the orientation range. It may also provide a better force distribution within the wrist, and also eliminate joint clearance via an overconstrained design.

Acknowledgements Dr. Wu is financially supported by the 2007–2013 ERDF Emilia-Romagna Regional Operational Programme at the Interdepartmental Center for Health Sciences and Technologies. Dr. Carricato gratefully acknowledges the financial support of the Italian Ministry of Education, Universities and Research through the PRIN grant No. 20124SMZ88. This work is also in partial fulfillment to Hong Kong RGC Grant No.616509, No.615610 and No.615212, China National Natural Science Foundation Grant No.51375413 and Shenzhen Municipal Science and Technology Planning Program for Basic Research No.JCYJ20120618150931822, which supported Dr. Wu when he was working at HKUST.

References

1. Rosheim, M.E.: Robot Wrist Actuators. Wiley, New York (1989)
2. Paul, R.P., Stevenson, C.N.: Kinematics of robot wrists. Int. J. Robot. Res. **2**(1), 31–38 (1983)
3. Bonev, I.A., Zlatanov, D., Gosselin, C.M.: Advantages of the modified Euler angles in the design and control of PKMs. In: Parallel Kinematic Machines International Conference, pp. 171–188. Chemnitz, Germany (2002). Accessed 23–25 April 2002
4. Gottlieb, D.H.: Robots and fibre bundles. Bull. Belg. Soc. Math. **38**, 219–223 (1986)
5. Conconi, M., Carricato, M.: A new assessment of singularities of parallel kinematic chains. IEEE Trans. Robot. **25**(4), 757–770 (2009)
6. Hayes, M.J.D., Husty, M.L., Zsombor-Murray, P.J.: Singular configurations of wrist-partitioned 6R serial robots: a geometric perspective for users. Trans. Can. Soc. Mech. Eng. **26**(1), 41–55 (2002)
7. Nubiola, A., Bonev, I.A.: Absolute calibration of an ABB IRB 1600 robot using a laser tracker. Robot. Comput.-Integr. Manuf. **29**(1), 236–245 (2013)
8. Gosselin, C.M.: The agile eye: a high-performance three-degree-of-freedom camera-orienting device. In: IEEE International Conference on Robotics and Automation (ICRA), pp. 781–786. San Diego, CA, USA (1994). Accessed 8–13 May 1994
9. Gosselin, C.M., Caron, F.: Two degree-of-freedom spherical orienting device. US Patent 5,966,991 (1999). Accessed 19 Oct 1999
10. Rosheim, M.E., Sauter, G.F.: New high-angulation omni-directional sensor mount. In: International Symposium on Optical Science and Technology, pp. 163–174. Seattle, WA, USA (2002). Accessed 7 July 2002
11. Culver, I.H.: Constant velocity universal joint. US Patent 3,477,249 (1969). Accessed 11 Nov 1969
12. Hunt, K.H.: Constant-velocity shaft couplings: a general theory. J. Manuf. Sci. Eng. **95**(2), 455–464 (1973)
13. Kong, X.W., Gosselin, C.M.: Type Synthesis of Parallel Mechanisms. Springer, Berlin (2007)

14. Wu, Y.Q., Li, Z.X., Shi, J.B.: Geometric properties of zero-torsion parallel kinematics machines. In: IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp. 2307–2312. Taipei, Taiwan (2010). Accessed 18–22 Oct 2010
15. Carricato, M.: Decoupled and homokinetic transmission of rotational motion via constant-velocity joints in closed-chain orientational manipulators. ASME J. Mech. Robot. **1**(4), 041008 (2009)
16. Hunt, K.H.: Kinematic Geometry of Mechanisms. Oxford University Press, Oxford (1978)
17. Wu, Y.Q., Liu, G.F., Löwe, H., Li, Z.X.: Exponential submanifolds: a new kinematic model for mechanism analysis and synthesis. In: IEEE/RSJ International Conference on Robotics and Automation (ICRA), pp. 4177–4182. Karlsruhe, Germany (2013). Accessed 6–10 May 2013
18. Wu, Y.Q., Löwe, H., Carricato, M., Li, Z.X.: Inversion symmetry of the euclidean group: Theory and application in robot kinematics. submitted to IEEE Trans. Robot. (2014). Accessed 22 Sept 2014
19. Löwe, H., Wu, Y.Q., Carricato, M.: Symmetric subspaces of SE(3). To appear in Adv. Geom. (2015)
20. Loos, O.: Symmetric Spaces. Benjamin, New York (1969)
21. Wu, Y.Q., Carricato, M.: Identification and geometric characterization of Lie triple screw systems. 14th IFTOMM World Congress, Taipei, Taiwan (2015). Accessed 25–30 Oct 2015
22. Murray, R.M., Li, Z.X., Sastry, S.S.: A Mathematical Introduction to Robotic Manipulation. CRC Press, Boca Raton (1994)
23. Meng, J., Liu, G.F., Li, Z.X.: A geometric theory for analysis and synthesis of sub-6 DoF parallel manipulators. IEEE Trans. Robot. **23**(4), 625–649 (2007)
24. Thompson, G.A.: Constant velocity coupling and control system therefor. US Patent 7,144,326 (2006). Accessed 5 Dec 2006
25. Kocabas, H.: Design and analysis of a spherical constant velocity coupling mechanism. ASME J. Mech. Design **129**(9), 991–998 (2007)
26. Ling, S.H., Huang, M.Z.: Kinestatic analysis of general parallel manipulators. ASME J. Mech. Design **117**(4), 601–606 (1995)

A Taxonomy of Benchmark Tasks for Robot Manipulation

Ana Huamán Quispe, Heni Ben Amor and Henrik I. Christensen

1 Introduction

A common theme among recent workshops and symposia on grasping and manipulation, is that there exists an "increasing difficulty in comparing the practical applicability of all new algorithms and hardware" [47]. Among the reasons that have led to this impasse are (a) the lack of standardized benchmark tasks, (b) a lack of suitable metrics to evaluate the integral performance of a robot, (c) the lack of comparative data that allows us to broadly categorize robot performance, and (d) the lack of guidelines for a realistic experimental setup. Hence, the methodology used to perform and evaluate robot manipulation experiments greatly varies in different publications, often rendering a comparison infeasible. Under these circumstances it is difficult to document and evaluate the progress made in the development of robot manipulation skills over the years. Although during the last years, important efforts have been made to evaluate robotic performance, there is still progress to be made (Fig. 1).

In this paper, we lay groundwork to formally establish a taxonomy of benchmark tasks for bimanual manipulators. We review literature concerning dexterity and functional capacity tests developed for medical purposes and extract guidelines to define clear methodologies to evaluate manipulation tasks. We also analyze the existing efforts towards benchmarking in the robotics literature and determine the commonalities, advantages and shortcomings among them. Our goal is to discuss both the possibility of standardizing robot manipulation benchmarks, as well as the

A. Huamán Quispe (✉) · H. Ben Amor · H.I. Christensen
Institute for Robotics and Intelligent Machines, Georgia Institute of Technology, Atlanta, GA 30332, USA
e-mail: ahuaman3@gatech.edu

H. Ben Amor
e-mail: hbenamor@cc.gatech.edu

H.I. Christensen
e-mail: hic@cc.gatech.edu

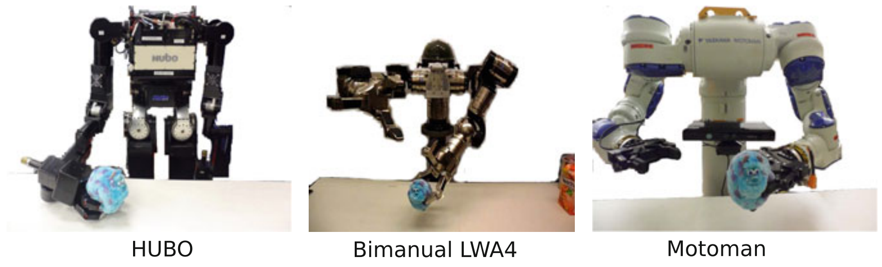


Fig. 1 How to compare different robots performing the same manipulation task?

difficulties that are hampering these efforts. We (1) propose a taxonomy of benchmark manipulation tasks and its corresponding metrics, (2) justify the need for evaluation frameworks that consider the robot as a whole and not as the sum of its parts, (3) discuss the strong need of realistic assumptions for robot experiments.

Our work focuses on service robots, although some of the sections can be considered useful for industrial robot evaluation as well. The rest of this paper is structured as follows: Sect. 2 presents a review of existing work in manipulation tests in both robotics and related fields. Section 3 presents the benchmark taxonomy proposed and examples of representative tasks and a brief discussion of each type. In this section we also discuss the guidelines proposed for realistic testing. Finally, Sect. 4 presents our concluding remarks.

2 Revisiting Manipulation Tests

Arguably, there is a deep relationship between human intelligence and dexterous manipulation. In [44], Williams presented experimental evidence showing that manual ability is highly correlated with the level of independence in elderly population. In this context, manual ability refers to the ability to perform actions involving object manipulation in order to achieve a goal. We will refer to this concept as hand function.

Definition 2.1 (*Hand Function*) The ability to use the hand(s) in everyday activities, which involves dexterity, manipulation skills and task performance skills.

Hand function then refers to using the hand(s) to accomplish a useful task, such as writing, cutting meat, pouring a drink or opening a door. All the tasks mentioned require dexterity, defined loosely here as the ability to manipulate an object with the hand. Dexterity can further be described by two related terms:

Definition 2.2 (*Manual Dexterity*) The ability to make skillful, well directed arm-hand movements in manipulating fairly large objects under speed conditions.

Definition 2.3 (*Fine-Motor Dexterity*) The ability to make rapid, skillful, controlled movements of small objects where the fingers are primarily involved.

As it was mentioned in Definition 2.1, hand function requires no one but a combination of multiple abilities such as motor, perceptual and cognitive skills, which constantly interact with each other.

Definition 2.4 (*Manipulation Skills*)
Motor Skills: Ability to actuate in the environment.
Perceptual Skills: Ability to gather information from the environment.
Cognitive Skills: Capacity to elaborate plans to achieve a goal, taking into account both motor and perceptual knowledge.

There has been acute interest in developing tests to evaluate the manipulation capabilities of both humans and robots. Regarding humans, testing is important in order, for example, to investigate the existence and degree of impairments caused by an injury, accident, or illness. In robotics, evaluation is important in order to compare quantitatively different research approaches.

Designing a unique test to evaluate hand function for robots (and for humans) is hard. As it was explained, manipulating an object involves the interaction of diverse, well-defined capabilities. Should these components be evaluated per separate? What metrics should be used? How should the tests be selected?

In the following section we analyze the most relevant tests we found in literature related to hand evaluation from a purely human perspective. Afterwards, we review robotics literature regarding benchmarking and common manipulation tasks performed by physical robots. We finish this section summarizing the similarities, differences and lessons learned from both types of tests.

2.1 Manipulation Tests for Humans

There is not an universal definitive test for hand function, hence diverse manipulation tests have appeared during the last 200 years under different names, such as hand function tests, dexterity tests and motor assessment evaluations. We reviewed 18 representative tests (Table 1). The procedure to select these tests was the following:

- Tests appearing in recent Physical Therapy surveys [27, 38] and whose original sources were available in English.
- Tests referenced by the ones above.
- Tests found through Google Scholar using the following search keywords: *manipulation, dexterity test, hand* and *function*.

Table 1 Reviewed manipulation tests

Test	Aspect measured
1. Kapanji test [21]	Thumb opposition
2. Hand strength [29]	Grip and pinch strength
3. Box and blocks dexterity test (BBT) [30]	Manual dexterity
4. Moberg pickup test (MPT) [35]	Finger dexterity & Sensory assessment
5. Nine-hole peg test (NHPT) [36]	Finger dexterity
6. Functional dexterity test (FDT) [1]	Finger dexterity
7. Purdue pegboard test (PPT) [31]	Finger dexterity
8. Minnesota rate of manipulation test (MRMT) [15]	Manual dexterity
9. Grooved pegboard test (GPT) [5]	Visuomotor aptitude & Speed
10. Jebsen hand function test (JHFT) [20]	Hand function
11. Sequential occupational dexterity assessment (SODA) [43]	Hand function
12. Wolf motor functional test (WMFT) [46]	Motion dexterity & Hand function
13. Chedoke arm and hand activity inventory (CAHAI) [3]	Hand function
14. Action research arm test (ARAT) [48]	Arm dexterity
15. Sollerman hand function test [40]	Hand function
16. Upper extremity performance test for the elderly (TEMPA) [10]	Hand function
17. Southampton hand assessment procedure (SHAP) [26]	Hand function
18. Arm motor ability test (AMAT) [23]	Hand function

Due to space constraints, we only give a general overview of some of the tests on Table 1 (A more detailed description can be found in our supplementary material[1]).

2.1.1 Manipulation Tests Through the Years

Pioneering studies on hand function focused on motor skills with emphasis on hand strength and joint mobility. Regarding the former, Mathiowetz et al. [29] analyzed 4 measures of strength: grip, palmar pinch, key pinch and tip pinch. Regarding the latter, in [21] the Kapandji Thumb Opposition scale is presented, which measures the range of motion of the hand by evaluating if the thumb is able to reach the other 4 fingers at their fingertips and at their joints.

The justification to use motor-skill tests was the high correlation between these skills and hand function. However, it was noted that these tests do not involve interaction with objects, which is a fundamental part of manipulation. This fact inspired

[1]www.cc.gatech.edu/~ahuaman3/docs/papers/supplementaryMaterial.pdf.

the development of tests involving simple manipulation actions, such as pick-and-place operations. In [30] Mathiowetz proposed the Block and Box Test (BBT) to measure manual dexterity. The test consists of picking up wooden blocks from a bin and throwing them into another bin as fast as possible. Another test, used for selective evaluation of assembly workers, is the Purdue Pegboard Test (PPT) [31], which requires the use of a board with rows of holes. The task consists on placing thin pegs into the holes with the left arm, then the right arm and then both arms at the same time. The test also included a final bimanual assembly task, where after placing the peg, 3 small objects are placed on top of it (two washers and a collar). In this case, the test emphasizes fine-motor dexterity.

The tests mentioned above involve the manipulation of pegs or cubes. Given the extensive variety of objects to be potentially grasped, additional tests were proposed to evaluate the ability of the hand to perform different grasps for objects with different geometry (cylindrical, spherical, three-jaw). An example of these is the ARAT test [48], composed of 4 sections, 3 of them related to grasping, gripping and pinching objects such as a wood cube, a cricket ball, a small metallic tube and a marble.

So far, the test reviewed are informative. However, it has been questioned if they truly measure the ability of the subjects to use their hand functionally. Take as an example stroke patients. It has been observed that, although many of them performed poorly in fine-dexterity tests, they nonetheless are capable of performing ordinary everyday tasks. The challenges in motor skills are faced with a better use of their cognitive skills. Given this, tests featuring functional tasks were introduced. The Jebsen Hand Function Test (JHFT) [20] consists on 7 tasks chosen as representative of common activities of daily living such as picking up cans, scooping beans with a spoon (simulated feeding) and stacking checkers. All of these tasks are unimanual. However, most manipulation tasks involve the use of two arms. Successive tests took care of including both type of tasks. In [40], the Sollerman test was proposed, consists of 20 tasks from which 14 are bimanual. An interesting aspect of this test is that the tasks were chosen such that the grasps used were proportional to their frequency. Recently, the SHAP test [24], inspired heavily by the Sollerman test, has been presented. This test is particularly interesting because it present tasks for both dexterity and functional evaluation (Fig. 2).

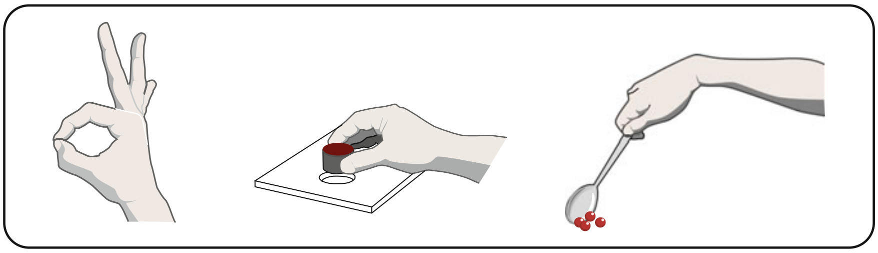


Fig. 2 Different samples of manipulation tests items used in occupational therapy: *Left* Kapandji test. *Middle* Peg-and-board generic test. *Right* Jebsen hand function test

2.2 Manipulation Tests for Robots

In this section we review manipulation tests from a robotics perspective. Similar to the case of human tests, there is not an evaluation tool unanimously embraced by the community. As a result, the research literature shows a myriad of robotic systems capable of executing heterogeneous tasks under varying assumptions. During the last years, though, there has been a growing interest in designing common tasks to compare the performance of different robot systems. Most of the existing tests for robot manipulators can be classified in two types, namely (1) **Component Benchmarking** and (2) **System Benchmarking**. Component benchmarking evaluates a single component of the robot system, while system benchmarking tests evaluate the robot system as a whole, considering the interactions between its components. In this section we only address tests of the second type.

In 1985, Collins proposed a test for industrial manipulators, commonly known as the Cranfield Assembly Benchmark [7], which consists on assembling 17 parts to form a mechanical pendulum. As it was noted in the paper, the goal of this test was *to examine the software and control features applied in the assembly process*. Given the structured nature of the assembly environment, perception was not considered as an element to be evaluated.

Home environments pose challenges far different from industrial settings: Each house is different, there exists many different household tasks, and each of them can potentially require a different set of skills from the robot. Therefore, it should not come as a surprise that designing a set of benchmark tasks is considered a daunting task. Nonetheless, through the years researchers have used common sense to choose tasks to evaluate their systems, such as window cleaning, fetching and pouring drinks and folding laundry, to name a few. Table 2 shows a sample of tasks currently performed by state-of-the-art bimanual manipulators along with the inherent assumptions.

With the advent of both affordable robotic manipulators and RGB-D sensors, the interest in benchmark tasks to compare robot performance was spurred. In 2008, Grunwald et al. [17] presented a set of benchmark tasks for dual-hand manipulators, as part of the DEXMART project [39]. The tests were targeted to bimanual manipulators and were chosen considering a cafeteria environment, so tasks such as carrying a tray with two hands and clearing a table were proposed.

Table 2 Common tasks presented in robots

Task	Robotic platform	Assumption
Making coffee	Rollin Justin [25]	3D model of objects is known
Capping a pen	Golem Crichton [8]	3D model of object is known
Picking up a human	RIBA [34]	Constant human guidance
Insert a plug into a socket	PR2 [32]	Fiducial marker in plug

Table 3 Manipulation tasks in recent robot competitions

Event	Task	Event	Task
DARPA ARM I	1. Grasp 12 objects: radio, rock, ball, flashlight, hammer, case, floodlight, shovel, screwdriver	RoboCup @Home	1. Hand an object to a human
	2. Staple a stack of paper		2. Grasp a cup and a bottle
	3. Turn on flashlight		3. Pick up objects from shelf
	4. Open door (handle)		4. Pick up 5 unknown tabletop objects
	5. Unlock a door (key)	DEXMART	1. Empty a trash bin
	6. Drilling		2. Open a screw cap
	7. Hang up a phone		3. Solve a Rubick Cube
DARPA ARM II	1. Change a tire		4. Pour water into a glass
	2. Open a bag, extract pliers and cut a wire		5. Carry a box with two hands
			6. Insert a battery into a drill
DRC	1. Carry and connect fire hose	APC	1. Bin-picking a single object
	2. Open series of doors		2. Bin-picking a single object in light clutter
	3. Drive and exit utility vehicle		
	4. Locate and close leaking valves		

In order to foster research advancement, robotic competitions featuring manipulation tasks have notably flourished in the last decade. In 2010, DARPA organized the Autonomous Robotic Manipulation (ARM) program, a 4-year project aimed to develop software and hardware to enable a robot to autonomously manipulate, grasp and perform complicated tasks with humans providing only high-level supervision. More recently, the Amazon Picking Challenge (APC) [41] has been proposed and aims to evaluate bin-picking skills in an industrial environment. Multiple other competitions exists that involve manipulation as one of the evaluated aspects. On 2013, the DARPA Robotics Challenge (DRC) took place in Miami and 4 out of its 8 tasks directly involved manipulation. A more traditional competition, the Robocup@Home [45], has been running since 2005. Its main goal is to evaluate the progress of mobile manipulators in a home environment. Navigation, manipulation and human-robot interaction were among other features evaluated. Table 3 shows the details of the tasks involved in the competitions mentioned above.

In order to define benchmark tasks, we must first define the methodology to follow. Efforts to establish standard practices for benchmarking manipulation have been shown by different authors however no clear answers have been delineated. The closest attempt, in our view, to ground clear protocols for manipulation benchmarks is the work presented in [19], in which the author suggests that benchmarking tests should be defined by two aspects: *Test Description*, which specifies the conditions under which the tests will be performed; and *Test Evaluation*, which describes how the task performance will be measured. Both of these aspects will be discussed in Sects. 2.3 and 2.4.

2.3 Test Evaluation

In order to compare reported results in robot manipulation, well-defined evaluation metrics are needed. Evaluation metrics can be of qualitative or quantitative nature and can address different sub-topics of manipulation. Robot hands and arms are typically first assessed through their physical characteristics and low-level performance indicators, e.g. degrees-of-freedom, applicable forces, number of fingers, etc. However, these performance indicators can not be used to infer better or worse manipulation capabilities, since this depends on the design of the hand, as well as the interplay of the components when put to test. In [9, 16] *taxonomies* are used to evaluate the range of executable grasps. Given a grasp taxonomy and a specific robot hand, the set of executable grasps can be identified and compared to other robots. To assess the range of grasps in a more quantitative manner, Feix et al. [13] introduced a so-called anthropomorphism index which compares reachable fingertip poses between a human hand and a robot hand. An *anthropomorphism index* is a continuous variable and can therefore be used to assess even small changes in the hand. Yet, it only compares feasible hand shapes and does not include contacts with objects. In contrast to that, the *graspability map* introduced in [37] explicitly analyses the contacts between a hand an object. A graspability map represents the set of poses that might lead to a precision force closure grasp on an object. Comparing the size and quality of graspability maps for different robot hands can indicate which hand is more likely to produce stable precision grasps.

The above performance metrics mainly target the mechanical design and hardware properties of a robot manipulator. Grasp planning algorithms, in contrast, are compared using a different set of metrics. A common approach is to evaluate the grasp quality using specific grasp quality metrics [4], such as the ϵ-metric, which measures the total and maximum finger force [14]. Yet, as was noticed in recent publications [2], grasp quality measures are typically calculated in simulations and often do not reflect the grasp executed by a robot. Hence, the most prominent approach for evaluations is to analyse the success rate in an object lifting task. Typically, a set of representative objects is grasped, then lifted, and the number of successful trials is documented. While such lifting tests are more informative, they still do not provide complete information about the degree of stability of the grasp. In [22] Kim et al.

suggested using both visual inspection, as well as interactive, physical inspection to evaluate the success of grasps. In interactive inspection, a human subject touched the object while it was grabbed by the robot, applied physical perturbations (jiggling it), and then evaluated its stability. In a similar vein, Morales et al. [33] used shaking movements after grasping in order to estimate the stability of a grasp.

While grasp stability is *necessary* for many tasks, it is not a *sufficient* condition for manipulation tasks. Many tasks that go beyond pick-and-place require additional dexterous capabilities. Benchmarking of manipulation skills has therefore moved towards *functional tests*, such as opening a series of doors, removing a screw cap, or inserting a battery into a drill. Functional tests can evaluated by assessing whether the complete task was successfully achieved and by counting the success of individual sub-tasks. Table 3 contains a list of functional tasks that have recently been used in major competitions and projects on grasping and manipulation. The simple interpretation of achieved results and the embedding in real-world scenarios, makes functional tests particularly appealing for robot competitions. In addition, functional tests do not come with an inherent assumption about the robot hardware. In contrast to the Kapanji test, grasp quality measures or taxonomic evaluations of dexterity, functional tests do not assume a specific morphology and can be performed with a wide range of different manipulators, e.g. jamming grippers, deformable hands, or anthropomorphic hands.

2.4 Test Description

In practice, it is often challenging to compare achieved manipulation performance with reported values in the literature, or even replicate a result reported in a different paper. Comparison of robot experiments can only be reasonably performed, if the involved tests are conducted under the same or sufficiently similar conditions. To overcome this challenge, we can design benchmarks in such a way as to minimize the variance in the inherent assumptions.

One approach to do this, is by clearly specifying *reasonable* assumptions. For example, as done in recent competitions, e.g. the Amazon Picking Challenge, specific benchmark objects along with their 3D models can be provided. Given a restricted set of objects to manipulate, it is also possible to provide a shared software framework for object detection. This would reduce the effect of perception on the manipulation benchmark. However, it also bears the risk of *over-specialization*; the design of very specific, brittle solutions that do not generalize to new situations.

Another approach to minimize variability is to use a standard robot hardware and software platform. For example, various publication on manipulation use the PR2 robot in conjunction with ROS for manipulation. This allows different research teams to share algorithms at the code level, thereby facilitating benchmarking different manipulation methods. However, standardizing the robot platform also limits the range of possible research questions that can be addressed. In the case of the PR2, for example, no in-hand manipulation can be studied.

In this paper, we propose two different measures to ensure a fair comparison of grasping and manipulation methods. The first measure we propose is the inclusion of **inherent stochasticity** into grasping benchmarks. Instead of specifying a set of conditions under which an experiment is performed, e.g. a set of locations and orientations of the object at the start of the experiment, we can design our benchmarks such that the conditions are determined by the stochasticity in the task. In the case of grasping and lifting an object, for example, we can introduce stochasticity by repeatedly dropping the object into a bin and then performing the task again. The configuration of the object in consecutive trials will depend on the resting position after dropping. As a result the variation in pose will be imposed by the task rather than a human expert. The second measure we propose for fair comparisons of manipulation capabilities, is the focus on longterm **multi-step** evaluations. Instead of executing a task only one time after which the human tester resets the environment, we run the experiment in a loop without human intervention. In the case of the above lifting example, we can have the robot repeatedly lift an object from a bin and throw it into a second bin in alternation. Since no human intervention is allowed, these tests capture the robot's ability to repeatedly deal with stochasticity inherent to these tasks.

3 Taxonomy of Benchmark Tasks

Section 2 provided an overview of the current status of benchmarking as well as insights to specify relevant test descriptions and test evaluations. In this section we present our proposed taxonomy of benchmark tasks for robot manipulators. Several criteria was considered in order to design the high-level taxonomy and the sample tasks that will be presented. In the following lines we will summarize these considerations, which we consider vital for a proper methodology of benchmark design:

1. **Hardware-agnostic**: Tests should not be designed with a specific platform in mind. In order for benchmarks to be widely accepted, they should be realizable under minimum hardware capabilities assumptions.
2. **Flexibility**: A lesson learned from the review of human tests is that there is not an unique "right way" to solve a task. The robot should be allowed to apply a strategy that better suits its particular situation (i.e. a robot equipped with a gripper might have to adopt a different approach to grasping an object than a robot with a 3-fingered hand).
3. **Time efficiency**: Benchmarking is not a goal by itself. Rather, it should be seen as a diagnostic tool to be applied regularly, to make sure our systems are comparable (or within reasonable distance) to the state-of-the-art. Accordingly, since benchmarks are a sidestep tool, they should be selected such that they can be evaluated in a rapid manner, without the need of a complex setup or highly constrained rules.
4. **Relevant metrics**: A metric is only useful if it can be compared against a standard value. Tasks should be selected such that the evaluation can be objective,

numerically expressed and important for both the researcher and, eventually, for the end-user. From the discussion in Sect. 2.3 we believe that (1) Task completion time and (2) Success rate are the two objective, informative metrics that can more easily be used.

5. **Statistically relevant**: Evaluation should consider a minimum number of attempts to be considered valid. This has already been seen in the ARM project evaluations, in which 5 trials were performed and the average results were evaluated.
6. **Realistic assumptions**: While a Laboratory environment is not the same as a real-home space, we should stress the importance of avoiding to rely on assumptions that in no way will exist in the real world. For instance, assumption of markers placed in objects or off-board visual sensors are very unlikely to occur, so their use might not translate into a realistic evaluation of capabilities.
7. **User-focused tests**: Service robots will be deployed at human homes, so it is reasonable to take into account feedback from the end-users to evaluate our systems. Studies in assistive technologies have shown that the main objective performance measures used by humans with assistive robotic arms are: Their capacity to perform activities of daily living and time to task completion [42].

Based on the reasons exposed, we designed the taxonomy of benchmark tests shown in Fig. 3. Details of each level in the classification follows.

3.1 Physical Tests

The tests in this section measure fundamental motor skills in a robot system, requiring very limited or null perceptual and cognitive skills. The purpose of this battery of tests is to evaluate the potential of the robot's hardware, which is related to its ability to solve a task. We further divide these tests in two types explained below. Examples of representative tasks are shown in Table 4.

Hand Tests

Tests that measure exclusively the hand physical capabilities. Given that a hand itself is often a complex system on its own, this decision seems reasonable. An important observation for this section is that these tests effectively evaluate the hand potential,

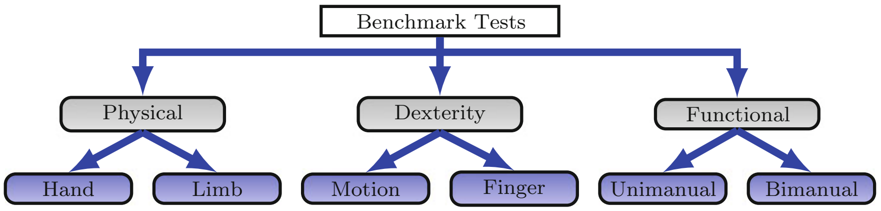


Fig. 3 Taxonomy of benchmark tasks

Table 4 Sample physical tasks

Sublevel	Examples	Sublevel	Examples
Hand	1. Maximum finger aperture	Limb	1. Position the palm on a table surface
	2. Maximum payload when picking up a high-friction object		2. Open a screw cap
	3. Perform static grasps from existing taxonomies		3. Point at diverse objects on a table
	4. Perform static grasps of benchmark objects [6, 28]		4. Sequentially point at a moving target

rather than the hand's actual ability to manipulate, which requires the additional evaluation of both perception and cognitive skills jointly with the motor skills. An example of a hand test commonly found in robotics literature is the use of grasp taxonomies for hand evaluation [9].

Limb Tests

Tests that measure capabilities of both hand and arm interacting together. Evaluation of simple online control is tested, such as capability to follow a workspace trajectory and capability to follow an end-effector pose. These abilities can be considered basic building blocks for the tests in the following sections.

3.2 Dexterity Tests

This type of tests evaluate motor skills, moderate perceptual skills and basic cognitive skills to solve simple tasks involving picking-up an object from a table, in-hand reconfiguration, or hand-eye coordination. A further subdivision of such tests is explained below.

Manual Dexterity Test

This type of tests measure the robot's ability to manipulate objects mainly for transport operations, in which in-hand manipulation is not required. In [11], Feix et al. found out - after analyzing video data of daily activities of 2 household workers and 2 machinists - that for many cases, power grasps were more vastly used instead of precision grasps, even for small objects. Hence, the geometry of the object play less a factor in the grasp selection when the task is simply pick and place. In other words, for this type of test tasks, an accurate 3D model should not be required.

Another interesting result from [12] is the fact that most of the objects (used by the subjects in the video) presented characteristics that made them easy to grasp:

Table 5 Sample dexterity tests

Sublevel	Examples	Sublevel	Examples
Manual	1. Pick-up an object on a clear table	Finger	1. Unscrew a bottle using fingers only
	2. Pick an object from a table and place it on a cupboard		2. Rotate a chopstick
	3. Pick an object from a box and set it at an adjacent box		3. Grab a short cylinder from the table and rotate it such that its supporting face ends up facing upwards.

Object mass was normally less than 500 g and the grasps required less than 7 cm in width. This information is corroborated by previous studies reporting the most common characteristics of manipulated objects [6, 28].

Fine-Motor Dexterity Test

In contrast to the manual tests, the fine-motor dexterity tests evaluate in-hand manipulation: The ability to modify an object configuration without using arm's movement. Fine-motor skills are challenging since they usually require more sophisticated sensing capabilities, such as tactile. Also, perceptual errors, which can be more or less tolerated and corrected when manipulating regular objects, can affect more adversely in this case.

The following table shows some sample tasks. In the manual examples, *object* is a generic term referring to objects of generic geometry (cylinder, cube, sphere) and weight no bigger than 500 g (Table 5).

3.3 Functional Tests

The tests in this section require the full interaction of motor, perceptual and cognitive skills in order to solve the task at hand, which presents the following characteristics:

- Functional tasks usually require more than one step to be accomplished, hence task planning at the cognitive level should be addressed.
- Task-specific constraints must be considered. Of particular importance are object's pose constraints.
- When faced with a household task, humans possess knowledge from previous experience. In particular, object information is usually available at some abstract level. Given this, and following the inspiration of [18], it is acceptable to assume that the robot have previous available knowledge regarding the object and how it can be used to solve the task.

Table 6 Sample functional tests

Sublevel	Examples	Sublevel	Examples
Unimanual	Open a door (handle/knob)	Bimanual	Cut a piece of Play-Doh on a table
	Plug in a power plug		Rotate a steering wheel 45 degrees
	Press level on an electric kettle		Empty a trash can (turn it upside down)
	Pick up a glass from a full dish rack		Pick up a tray with a glass on it and transport it
	Push an emergency button		Open a jar with screw lid
	Pour a liquid in a wide container		Grab an open tetrapak and pour liquid in a cup for 2 s.
	Stir slowly in a pot		
	Spray from a bottle		
	Stack cans		
	Spoon beans (simulated feeding)		

- In a similar manner to the previous consideration: Information should be available but not overly specific: Tests should evaluate reasonable generalization of abstract knowledge (i.e. ability to grasp similar objects).

In these tests we include tasks involving the use of one and both arms, as a good fraction of household tasks involve the interaction of 2 limbs actuating either on the same object or on two objects interacting on the same task (Table 6).

4 Conclusion

In this paper, we have presented a taxonomy of benchmark tasks for robot manipulators. Using as inspiration the lessons learned from robotics and related fields we analyzed and proposed basic concepts as groundwork to formally define a standard manipulation benchmark methodology with 3 general levels describing robot capabilities with increasing levels of complexity. We emphasized the vital importance of system evaluation in contrast to component-wise testing, since it allows a more realistic assessment of a robot functional abilities. Our next step, which is part of our ongoing work, is to implement more of our proposed tasks and to make our performance evaluation publicly available for comparative studies. As is the case in other fields, benchmarking is not an static process. Rather, it is an evolving procedure that is shaped, improved and redefined after constant evaluation. We think this is an exciting challenge for our community and a step towards achieving more capable robotic manipulation systems.

References

1. Aaron, D., Jansen, C.: Development of the functional dexterity test (FDT): construction, validity, reliability, and normative data. J. Hand Ther. **16**(1), 12–21 (2003)
2. Balasubramanian, R., Xu, L., Brook, P., Smith, J., Matsuoka, Y.: Physical human interactive guidance: identifying grasping principles from human-planned grasps. The Human Hand as an Inspiration for Robot Hand Development. Springer, Switzerland (2014)
3. Barreca, S., Gowland, C., Stratford, P., Huijbregts, M., Griffiths, J., Torresin, W., Dunkley, M., Miller, P., Masters, L.: Development of the chedoke arm and hand activity inventory: theoretical constructs, item generation, and selection. Topics Stroke Rehabil. **11**(4), 31–42 (2004)
4. Bohg, J., Morales, A., Asfour, T., Kragic, D.: Data-driven grasp synthesis—a survey. IEEE Trans. Robot. **30**, (2013)
5. Bryden, P., Roy, E.: A new method of administering the grooved pegboard test: performance as a function of handedness and sex. Brain Cogn. **58**(3), xxx (2005)
6. Choi, Y., Deyle, T., Chen, T., Glass, J., Kemp, C.: A list of household objects for robotic retrieval prioritized by people with ALS. In: IEEE International Conference on Rehabilitation Robotics (2009)
7. Collins, K., Palmer, A., Rathmill, K.: The development of a European benchmark for the comparison of assembly robot programming systems. Robot Technology and Applications. Springer, Heidelberg (1985)
8. Dantam, N., Ben Amor, H., Christensen, H., Stilman, M.: Online multi-camera registration for bimanual workspace trajectories. In: HUMANOIDS (2014)
9. Deimel, R., Brock, O.: A novel type of compliant, underactuated robotic hand for dexterous grasping. In: Proceedings of Robotics: Science and Systems, pp. 1687–1692 (2014)
10. Desrosiers, J., Hébert, R., Dutil, E., Bravo, G.: Development and reliability of an upper extremity function test for the elderly: the TEMPA. Can. J. Occup. Ther. **60**(1), 9–16 (1993)
11. Feix, T., Bullock, I., Dollar, A.: Analysis of human grasping behavior: correlating tasks, objects and grasps. IEEE Trans. Haptics **7**, 430–441 (2014)
12. Feix, T., Bullock, I., Dollar, A.: Analysis of human grasping behavior: object characteristics and grasp type. IEEE Trans. Haptics **7**, 311–323 (2014)
13. Feix, T., Romero, J., Ek, C., Schmiedmayer, H., Kragic, D.: A metric for comparing the anthropomorphic motion capability of artificial hands. IEEE Trans. Robot. **29**(1), 82–93 (2013)
14. Ferrari, C., Canny, J.: Planning optimal grasps. In: ICRA (1992)
15. Gloss, D., Wardle, M.: Use of the minnesota rate of manipulation test for disability evaluation. Percept. Mot. Skills **55**(2), 527–532 (1982)
16. Grebenstein, M.: The awiwi hand: an artificial hand for the DLR hand arm system. Approaching Human Performance, pp. 65–130. Springer, Switzerland (2014)
17. Grunwald, G., Borst, C., Zöllner, J.E.A.: Benchmarking dexterous dual-arm/hand robotic manipulation. In: IROS Workshop on Performance Evaluation and Benchmarking (2008)
18. Hackett, D., Pippine, J., Watson, A., Sullivan, C., Pratt, G.: An overview of the DARPA autonomous robotic manipulation (ARM) program. J. Robot. Soc. Jpn. **31**(4), 326–329 (2013)
19. Iossifidis, I., Lawitzky, G., Knoop, S., Zöllner, R.: Towards benchmarking of domestic robotic assistants. Advances in Human-Robot Interaction. Springer, Heidelberg (2005)
20. Jebsen, R., Taylor, N., Trieschmann, R., Trotter, M., Howard, L.: An objective and standardized test of hand function. Arch. Phys. Med. Rehabil. **50**(6), 311 (1969)
21. Kapandji, A.: Clinical test of apposition and counter-apposition of the thumb. Annales de chirurgie de la main: organe officiel des societes de chirurgie de la main **5**(1) (1985)

22. Kim, J., Iwamoto, K., Kuffner, J., Ota, Y., Pollard, N.: Physically based grasp quality evaluation under pose uncertainty. IEEE Trans. Robot. **29**, 1424 (2013)
23. Kopp, B., Kunkel, A., Flor, H., Platz, T., Rose, U., Mauritz, K., Gresser, K., McCulloch, K., Taub, E.: The arm motor ability test: reliability, validity, and sensitivity to change of an instrument for assessing disabilities in activities of daily living. Arch. Phys. Med. Rehabil. **78**(6), 615–620 (1997)
24. Kyberd, P., Murgia, A., Gasson, M., Tjerks, T., Metcalf, C., Chappell, P., Warwick, K., Lawson, S., Barnhill, T.: Case studies to demonstrate the range of applications of the Southampton Hand Assessment Procedure. Br. J. Occup. Ther. **72**(5), 212–218 (2009)
25. Leidner, D., Borst, C., Hirzinger, G.: Things are made for what they are: solving manipulation tasks by using functional object classes. In: HUMANOIDS (2012)
26. Light, C.M., Chappell, P.H., Kyberd, P.: Establishing a standardized clinical assessment tool of pathologic and prosthetic hand function: normative data, reliability, and validity. Arch. Phys. Med. Rehabil. **83**(6), 776–783 (2002)
27. Lin, S., Chang, J., Chen, P., Mao, H.: Hand function measures for burn patients: a literature review. Burns (J. Int. Soc. Burn Inj.) **39**(1), 16–23 (2013)
28. Matheus, K., Dollar, A.: Benchmarking grasping and manipulation: properties of the objects of daily living. In: IROS (2010)
29. Mathiowetz, V., Weber, K., Volland, G., Kashman, N.: Reliability and validity of grip and pinch strength evaluations. J. Hand Surg. **9**(2), 222–226 (1984)
30. Mathiowetz, V., Volland, G., Kashman, N., Weber, K.: Adult norms for the box and block test of manual dexterity. Am. J. Occup. Ther. **39**(6), 386–391 (1985)
31. Mathiowetz, V., Rogers, S., Dowe-Keval, M., Donahoe, L., Rennells, C.: The purdue pegboard: norms for 14-to 19-year-olds. Am. J. Occup. Ther. **40**(3), 174–179 (1986)
32. Meeussen, W., Wise, M., Glaser, S., Chitta, S., McGann, C., Mihelich, P., Marder-Eppstein, E., Muja, M., Eruhimov, V., Foote, T., et al.: Autonomous door opening and plugging in with a personal robot. In: ICRA, pp. 729–736 (2010)
33. Morales, A., Chinellato, E., Sanz, P., Del Pobil, A., Fagg, A.H.: Learning to predict grasp reliability for a multifinger robot hand by using visual features. In: AISC (2004)
34. Mukai, T., Hirano, S., Nakashima, H., Kato, Y., Sakaida, Y., Guo, S., Hosoe, S.: Development of a nursing-care assistant robot RIBA that can lift a human in its arms. In: IROS (2010)
35. Ng, C., Ho, D., Chow, S.: The Moberg pickup test: results of testing with a standard protocol. J. Hand Ther. **12**(4), 309–312 (1999)
36. Poole, J., Burtner, P., Torres, T., McMullen, C., Markham, A., Marcum, M., Anderson, J., Qualls, C.: Measuring dexterity in children using the nine-hole peg test. J. Hand Ther. **18**(3), 348–351 (2005)
37. Roa, M., Hertkorn, K., Zacharias, F., Borst, C., Hirzinger, G.: Graspability map: a tool for evaluating grasp capabilities. In: IEEE/RSJ IROS, pp. 1768–1774 (2011)
38. Schoneveld, K., Wittink, H., Takken, T.: Clinimetric evaluation of measurement tools used in hand therapy to assess activity and participation. J. Hand Ther. **22**(3), 221–236 (2009)
39. Siciliano, B.: Advanced Bimanual Manipulation: Results from The DEXMART Project, vol. 80. Springer, Heidelberg (2012)
40. Sollerman, C., Ejeskär, A.: Sollerman hand function test: a standardised method and its use in tetraplegic patients. Scand. J. Plast. Reconstr. Surg. Hand Surg. **29**(2), 167–176 (1995)
41. The Amazon Picking Challenge. http://amazonpickingchallenge.org/ (2014)
42. Tsui, K., Feil-Seifer, D., Matarić, M.J., Yanco, H.: Performance evaluation methods for assistive robotic technology. Performance Evaluation and Benchmarking of Intelligent Systems, pp. 41–66. Springer, US (2009)
43. van Lankveld, W., van't Pad Bosch, P., Bakker, J., Terwindt, S., Franssen, M.: Sequential occupational dexterity assessment (SODA): a new test to measure hand disability. J. Hand Ther. **9**(1), 27–32 (1996)
44. Williams, M., Hadler, N., Earp, J.: Manual ability as a marker of dependency in geriatric women. J. Chronic Dis. **35**(2), 115–122 (1982)

45. Wisspeintner, T., Van Der Zant, T., Iocchi, L., Schiffer, S.: Robocup@Home: scientific competition and benchmarking for domestic service robots. Interact. Stud. **10**(3), 392–426 (2009)
46. Wolf, S., Thompson, P., Morris, D., D.K., Winstein, C., Taub, E., Giuliani, C., Pearson, S.: The EXCITE trial: attributes of the wolf motor function test in patients with subacute stroke. Neurorehabil. Neural Repair **19**(3), 194–205 (2005)
47. Workshop on Autonomous Grasping and Manipulation: An Open Challenge. http://grasping-challenge.org/ (2014)
48. Yozbatiran, N., Der-Yeghiaian, L., Cramer, S.: A standardized approach to performing the action research arm test. Neurorehabil. Neural Repair **22**(1), 78–90 (2008)

The Robotic Sixth Finger: A Wearable Compensatory Tool to Regain Grasping Capabilities in Paretic Hands

Gionata Salvietti, Irfan Hussain and Domenico Prattichizzo

1 Introduction

Wearable robots are expected to work very closely, to interact and collaborate with people in an intelligent environment [1]. Traditionally, wearable robotic structures have been mainly used in substitution of lost limbs (e.g., prosthetic limbs) or for human limb rehabilitation (e.g., exoskeletons). However, the progress in miniaturization and efficiency of the technological components is allowing more light and compact solutions, enhancing user's safety and comfort, while opening new opportunities for wearable robot use [2]. Together with exoskeleton and prosthesis, a very promising research direction seems to be that of adding robotic limbs to human, rather than substituting or enhancing them [3]. This addition could let the humans augment their abilities and could give support in everyday tasks to impaired people. This paper investigates how to compensate the capabilities of the human hand, instead of developing additional robotic extra-arms, as discussed for instance in [4]. The idea of using an extra-finger to support the human hand in grasping functions was initially proposed in [5]. Then, independently both in [6–8], the authors proposed the use of extra fingers to support the human hand to grasp objects whose size does not fit a hand or in executing bimanual tasks with one hand. The main difference is

G. Salvietti (✉) · I. Hussain · D. Prattichizzo
Dipartimento di Ingegneria Dell'Informazione E Scienze Matematiche,
Università degli Studi di Siena, Via Roma 56, 53100 Siena, Italy
e-mail: salviettigio@dii.unisi.it

I. Hussain
e-mail: hussain@dii.unisi.it

D. Prattichizzo
e-mail: domenico.prattichizzo@iit.it

D. Prattichizzo
Department of Advanced Robotics, Ististuto Italiano di Tecnologia, Via Morego 30,
Genova, Italy

A. Bicchi and W. Burgard (eds.), *Robotics Research*, Springer Proceedings
in Advanced Robotics 2, DOI 10.1007/978-3-319-51532-8_26

that in [7, 8], the goal was to minimize the size and the weight of the unique extra limb, while in [6], two extra fingers were used so to hold objects. While in [9] the authors developed a control strategy to grasp and manipulate objects, in [10] the authors mainly focused on the use of extra fingers for post-stroke patients.

Focusing on the hand, many wearable devices have been proposed in the last decade, especially for hand rehabilitation and function recovery. A review on robot-assisted approaches to motor neurorehabilitation can be found in [11]. In [12] the authors presented a comprehensive review of hand exoskeleton technologies for rehabilitation and assistive engineering, from basic hand biomechanics to actuator technologies.

However, most of the devices proposed in literature are designed either to increase the functional recovery in the first months of the rehabilitation therapy, when biological restoring and reorganization of the central nervous system take place, or are designed to augment human hand capabilities of healthy subjects by coordinating the device motion to that of the hand.

To the best of our knowledge, only few works target on the robotic compensation of hand function in the latter phase of rehabilitation. This means that patients usually after 6–9 months of rehabilitation must rely only on compensatory strategies by improving adaptations that increase the functional disparity between the impaired and the unaffected upper limb [13].

This work focuses on the compensation of hand function in patients with paretic limbs, e.g. chronic stroke patients. The final aim is to provide the patient with an additional robotic finger worn on the wrist. The Robotic Sixth Finger is used together with the paretic hand to seize an object, as shown in Fig. 1. The systems acts like a two-finger gripper, where one finger is represented by the Robotic Sixth Finger, while the other by the patient paretic limb. The proposed device goes beyond exoskeletons: it adds only what is needed to grasp, i.e. an extra thumb. We presented in [7] a preliminary version of a robotic extra-finger showing how this wearable device is

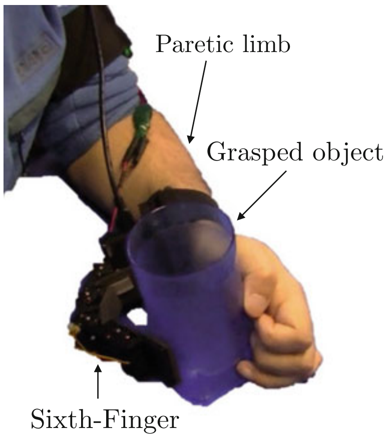


Fig. 1 The Robotic Sixth Finger. The device can be worn on the forearm thanks to an elastic band. When activated, it interacts with the paretic hand in grasping tasks

able to enhance grasping capabilities and hand dexterity in healthy subjects. In [8], we also presented an object-based mapping algorithm to control robotic extra-limbs without requiring explicit commands by the user. The main idea of the mapping was to track human hand by means of dataglove and reproduce the main motions on the extra-finger. This kind of approach is not suitable for patients with a paretic limb due to the reduced mobility of the hand. Therefore, we developed a wearable interface embedded in a ring to activate and use the finger [14].

In this work, we propose two possible designs of devices. The first model is a modular fully actuated finger. The other design consists of an underactuated finger which is compliant and consequently able to adapt to the different shapes of the objects.

For validation purposes, pilot experiments with two chronic stroke patients were performed. The experiments consisted in wearing the Robotic Sixth Finger and performing a rehabilitation test referred to as Frenchay Arm Test [10, 15]. Finally, we present preliminary results on the use of the extra-fingers for grasping objects for Activities of Daily Living (ADL).

2 Designs of the Robotic Sixth Finger

In this work, we propose three different solutions for the realization of a wearable extra-finger. The first is a fully actuated modular structure. The other two share a similar underactuated design. In the following, the three models are described in detail.

2.1 Fully Actuated Finger

The fully actuated finger consists of 1–DoF modules connected through screws. Modularity of the device offers two main advantages. Firstly, the length of the device and the number of actuated DoFs can be selected by choosing the total number of modules. Secondly, the robustness of the device is increased considering that robot parts are interchangeable. Each module consists of a servomotor (HS-53 Microservo, HiTech, Republic of Korea) and a 3D printed plastic part with an overall dimension of $42 \times 33 \times 16$ mm. The motor can provide a maximum torque of 1.5 kg cm. The modules can be connected in a pitch-pitch configuration, in order to replicate the flexion/extension motion of the human finger. The modular part of the finger is connected to a support base which contains also the electronic housing. A rubber band allows to easily wear the device on the forearm. The fully actuated finger is shown in Fig. 2. An external battery is used to provide power to all the circuits. All the electronics is enclosed in a 3D printed box attached to the finger to make it wearable. The module actuators are PWM controlled servomotors. The PWM signal is generated by an Arduino Nano board [16].

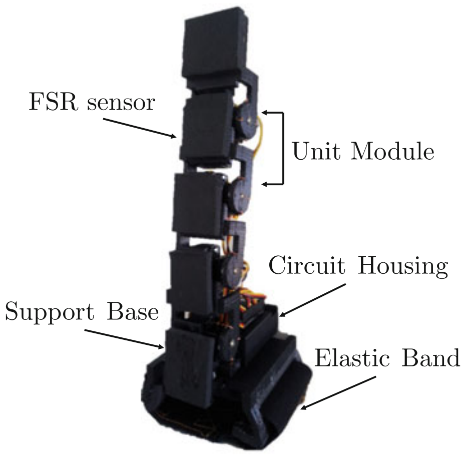


Fig. 2 The fully actuated Sixth-Finger. The finger is built using modules connected to a wrist elastic band

The user can command the finger motion by using the wearable switch placed on a ring, see Fig. 6. The switch is used to start the robotic finger flexion procedure and to move the finger back to an initial predefined position. The ring has been designed to be worn on the index finger of the non-paretic hand. In this way, the user can press the push button on the ring using his thumb.

We introduce a new control strategy that enables the finger to autonomously adapt to the shape of the grasped object. When the switch is activated, the finger starts to flex with a fixed joint angle increment, equal for each module, from a predefined position. We considered the completely extended finger as the starting position to enlarge the set of possible graspable objects. Each module has been equipped with a Force Sensing Resistor (FSR) (408, Interlink Electronics Inc., USA) able to detect contacts with the grasped object, see Fig. 2. As soon as one module is in contact with the object, that module stops its motion, while the others keep moving toward the object. During the grasping phase, the FSR sensors are in charge of detecting the contact of each module with the grasped object. In order to have suitable contact points, we set different closing priorities per each module. If the fingertip module comes in contact first, the remaining modules stop. If the second last comes in contact first, modules below to it stop, while the fingertip module keeps moving. The same methodology is followed for the other intermediate modules. Finally, if the module at the base of the finger is the first to come in contact, two different behaviors can occur: (i) the intermediate modules gets in contact before the fingertip; (ii) the fingertip module comes in contact first. In case (i) the fingertip is left free to move to get in contact with the object; in case (ii) the intermediate modules stop. The grasping procedure is commanded by the user acting on the switch. When the grasp is complete, the finger starts to autonomously keep the grasp stable. This grasp stabilization is obtained by controlling the compliance of each module.

Generally, in active compliance control framework, the motor torque is related to its position error through a stiffness constant [17]. The compliant (or stiff) behavior of the joint is achieved by virtue of the control, differently from what happens in mechanical systems with a prevalent dynamics of elastic type. This controller is typically used with actuators that can be torque controlled. In servomotors, it is not possible to directly command the torque to be exerted by the joints. Therefore, we modified the servomotors in order to read the joints position from the embedded potentiometer and we introduced a scaling factor k_d to modulate the displacement of the joints related to the applied forces. The compliance decreases with the increase of k_d. Details on how to vary the compliance of a module acting on servomotors control and simulating a variable stiffness can be find in [18].

The basic idea is that the module can change its compliance according to the force observed by each module through the relative FSR sensor. Thus, when the user pushes the object toward the extra finger to tight the grasp, the device becomes stiffer. The possibility to independently regulate each module's compliance allows to adapt the finger to the shape of the grasped object also during manipulation tasks. Similarly to what we did for the grasping procedure, we set the same priorities between the four modules also regarding the compliance variation.

When the user wants to release the grasped object, he just needs to lower the force exerted by his hand on the object and, automatically, the robotic device will make its joints more compliant. Eventually, by pressing again the switch, the robotic finger moves back to its home position by following a predefined trajectory. Note that if the patient is not able to exert force on the object, due to the motor deficit or to the position of the Robotic Sixth Finger on the forearm, the grasp tightness can be controlled through the wearable switch. The more the button on the ring is pressed, the more is the force exerted by the finger onto the object.

2.2 Underactuated Fingers

Underactuated robotic fingers are generally obtained using elastic elements in the design of their "unactuated" joints [19], which are usually passively driven [20]. The concept of underactuation in robotic fingers is different from that usually presented in robotic systems [21]. In an underactuated finger, the actuation force is applied to the input of the finger and is transmitted to the phalanges through a suitable mechanical design, e.g., four-bar linkages, pulleys and tendons, gears, etc. Since underactuated fingers have many DoFs, say n, and fewer than n actuators, passive elements are used to kinematically constrain the finger and ensure the shape adaptation of the finger to the grasped object [22].

We designed two underactuated Robotic Sixth Fingers to explore a passive compliance solution to the problem of adapting the robotic finger to the shape of the grasped object. The first version resembles the human finger shape. We placed in the support base a servomotor (HS-485 HB HITEC RCD Inc., USA) which takes care of the finger actuation. The motor has a stall torque of 6.0 kg cm and it is used

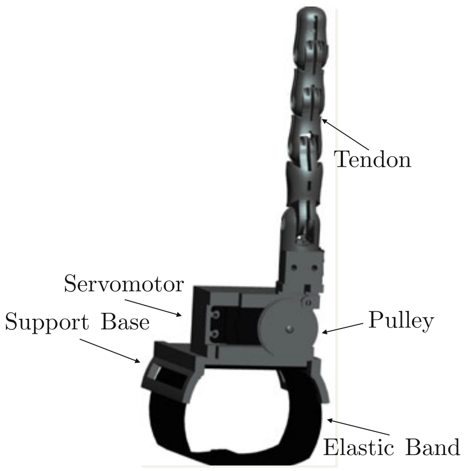


Fig. 3 Cad model of complete underactuated finger having servo motor with pulley. The rubber band is to wear the finger at wrist/arm

for the flexion/extension of the whole finger. Also this prototype can be worn on the forearm/wrist with the help of an elastic band. The finger's drive mechanism is a nylon wire, which connects the outermost phalanges with the motor through a pulley. The CAD model of the finger is reported in Fig. 3.

All the phalanges are connected by screws. Bearings are used to reduce the friction between phalanges. Also in this case we have pursued modularity and wearability concepts in designing the device. Modules can be added at the base of the finger structure depending on the required length of the finger.

If no torque is applied, the finger is completely straight. When the motor is activated, the wire is pulled and the finger bends to grab the object. The elastic rubber parts placed on the back of the finger between each phalanx are used to bring it to its original position when required. The control of the finger motion is simplified with respect to the fully actuated version, since compliance and shape adaptation are passively guaranteed by the finger structure. The user can regulate the finger flexion and the related applied force, acting on the push button. The more the button is kept pressed, the more is the flexion command through the servomotor.

In the second underactuated design proposed, we have duplicated the finger structure to improve the grasp robustness. We used different modules so to exploit the underactuation and the passive adaptation to the grasped object, while reducing the total weight of the device. So that, we considered modules that do not resemble the shape of the human finger. The actuation and design are inspired by recent works on underactuated robotic hands [19, 23]. Each module consists of a 3D printed polymeric part that acts as a rigid link and a 3D printed thermoplastic polyurethane part that realizes the flexible joint. Soft rubber pads are glued to the rigid links to increase the friction at possible contact areas. The modules can be linked by sliding the thermoplastic polyurethane part in the ABS part to speed up the assembling process. A cable is used to achieve the tendon driven actuation. The tendon wire runs through the two fingers and is attached to a lever rigidly connected to the actuator shaft. The

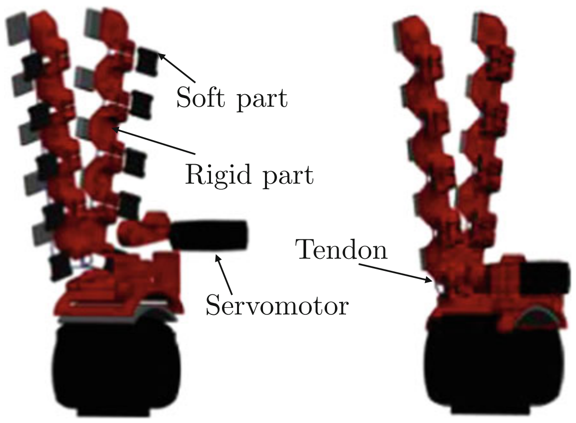


Fig. 4 The Double Robotic Sixth Finger. On the *left side*, the CAD exploded view of the device. On the *right side*, the CAD assembled view

Table 1 The Double Sixth Finger technical details

Module dimensions	4.2 × 2.0 × 0.9 cm
Module dimensions when connected	3.4 × 2.0 × 0.9 cm
Module weight	5.8 g
Support base dimensions	6.4 × 7.0 × 0.35 cm
Actuator control unit box dimensions	7.0 × 9.0 × 3.4 cm
Actuator control unit box weight	0.146 Kg
Actuator max torque	17 kg cm @ 12 V
Max. operating angles	0 – 180 deg
Max non-loaded velocity	5.1 rad/s
The Double-SoftSixthFinger total weight	0.166 kg

CAD exploded and assembled views are reported in Fig. 4. Technical details of the device are reported in Table 1.

2.3 *Positioning of the Devices on the Paretic Arm and Activation Interface*

We propose in this work three different prototypes of the Robotic Sixth Finger. All the designs share a common principle of work which consists in opposing to the paretic hand/wrist so to restrain the motion of the object. We will use the general term Robotic Sixth Finger when referring to features common to all the proposed devices. All the robotic extra-fingers can be worn in the distal part of the forearm (near, or on the wrist) since the grasp occurs by opposing the device to the paretic hand.

However, the distal positioning of the Robotic Sixth Finger may fail when the motor deficit is so advanced that a pathological synergism in flexion took place: in this case, the wrist becomes too much flexed and fingers are too much closed towards the palm to allow a successful grasping. When this pathological condition occurs, the Robotic Sixth Finger may be positioned more proximal at the forearm, in a way that the grasp can be achieved by the extra-finger opposition to the radial aspect of the thenar eminence. An example of two possible positions for the Robotic Sixth Finger are reported in Fig. 5. This flexibility in the positioning is achieved thanks to the modularity of the structures and the fixing support. Modularity allows to regulate the size and dexterity of the finger according to the position on the forearm and according to each patient's limb characteristics. The support base of the finger can be translated or rotated along the arm to place the finger in a suitable orientation. An elastic band and rubber spacers are used to increase the grip and comfort while reducing the fatigue during continuous use of the finger.

Concerning the easiness of the use, the patient can activate the robotic extra-finger motion through a switch. The switch is a push button placed on a ring worn on the healthy hand, see Fig. 6. This dramatically simplifies the interaction with the device. However, this simplification in the device command has to be compensated by an increase of the autonomy of the robotic prosthesis. In fact, the robotic finger needs to

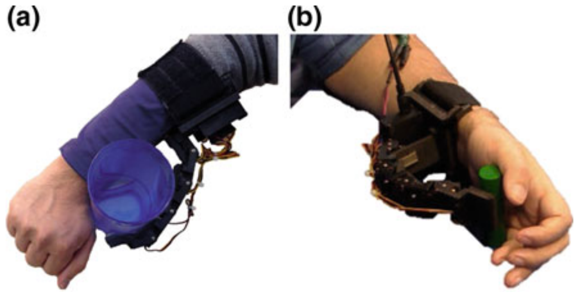


Fig. 5 The robotic extra finger in two possible configurations on the forearm. **a** The grasp is obtained at the wrist level. **b** The grasp is obtained at the hand level

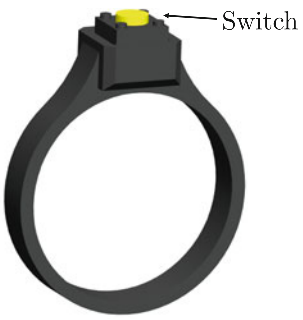


Fig. 6 The CAD model of the activating ring. The push button is used to start/stop the finger

adapt to the shape of the grasped object and it needs to be able to stabilize the grasp during the arm motion. We propose two different solutions to deal with these issues. The first is to use a fully actuated device which can actively regulate the compliance of its joints. The second, used in two of the proposed models, is to combine passive compliance with underactuation.

3 Pilot Experiment

In the current proof of concept study, we tested with two subjects how the Robotic Sixth Finger device can compensate for grasping capability. The aim was to verify the potential of the approach and to understand how the subjects can interact with the wearable device. In this direction, we performed a qualitative test, the Frenchay Arm Test [15]. The test consists of five pass/fail tasks to be executed in less then three minutes. The patient scores 1 for each of the successfully completed task, while he/she scores 0 in case of fail. The subject sits at a table with his hands in his lap, and each task starts from this position. He/she is then asked to use his/her affected arm/hand to:

1. *Task_1* Stabilize a ruler, while drawing a line with a pencil held in the other hand. To pass, the ruler must be held firmly.
2. *Task_2* Grasp a cylinder (12 mm diameter, 5 cm long), set on its side approximately 15 cm from the table edge, lift it about 30 cm and replace without dropping.
3. *Task_3* Pick up a glass, half full of water positioned about 15 to 30 cm from the edge of the table, drink some water and replace without spilling.[1]
4. *Task_4* Remove and replace a sprung clothes peg from a 10mm diameter dowel, 15 cm long set in a 10 cm base, 15 to 30 cm from table edge. Not to drop peg or knock dowel over.
5. *Task_5* Comb hair (or imitate); must comb across top, down the back and down each side of head.

The subjects taking part to the experiment have been affected by stroke more than two years before. The rehabilitation team have declared that no more functional improvements are achievable with respect to the gained upper limb motor performance. In particular, the patients showed the following characteristics based on the National Institute of Health Stroke Scale (NIHSS) [24]: (1) normal consciousness (NIHSS, item1a, 1b, 1c = 0), absence of conjugate eyes deviation (NIHSS, item 2 = 0), absence of complete hemianopia (NIHSS, item 3 $\leq$ 1), absence of ataxia (NIHSS, item 7 = 0), absence of completely sensory loss (NIHSS, item 8 $\leq$ 1), absence of aphasia (NIHSS, item 9 = 0), absence of profound extinction and inattention (NIHSS, item 11 $\leq$ 1). Patients received the Robotic Sixth Finger in the paretic hand, the left hand for one subject and the right one for the other. Thanks to the flexibility of the devices, the same prototypes were used in both subjects. For

[1]Note that for safety reasons we did not use water in presence of electronic components.

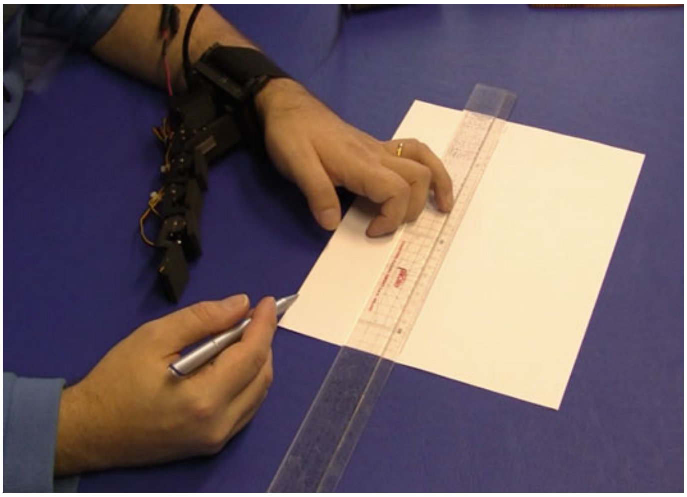


Fig. 7 Stabilize a ruler, while drawing a line with a pencil held in the other hand. The Sixth Finger does not interfere with the task execution

being included in the experiment, patients have shown partial paresis of the upper limb tested with the NIHSS item 5 “paretic arm” ≤ 2. Written informed consent was obtained from all participants. The procedures were in accordance with the Declaration of Helsinki.

The subjects performed the Frenchay Arm Test wearing in order the fully actuated finger, the underactuated human-like version, the double finger version and finally without any device. The Robotic Sixth Finger was placed on the paretic limb, while it was activated using the switch placed on the index of the other hand. Figure 7 reports a snapshot of the execution of *Task_1*, while Figs. 8 and 9 show the *Task_2* and *Task_3*, respectively.

The results of the tests are reported in Tables 2 and 3 for the two patients, respectively. Both the subjects were able to grasp a cylinder and to pick up a glass with the help of the robotic devices. In total, we got an improvement of 2 out of 5 points in the test scale.

3.1 The Robotic Sixth Finger in ADL

The main target of the Robotic Sixth Finger is that of giving to the users a compensatory tool that can be used in common ADL so to improve their quality of life. As a first example, we tested how the proposed devices could help the patients in an unstructured kitchen. In particular, the patients were asked to take advantage of the extra-fingers to open different cans and jars. These operations are typical bimanual tasks, where one hand is used to restrain the motion of the object, while the other is

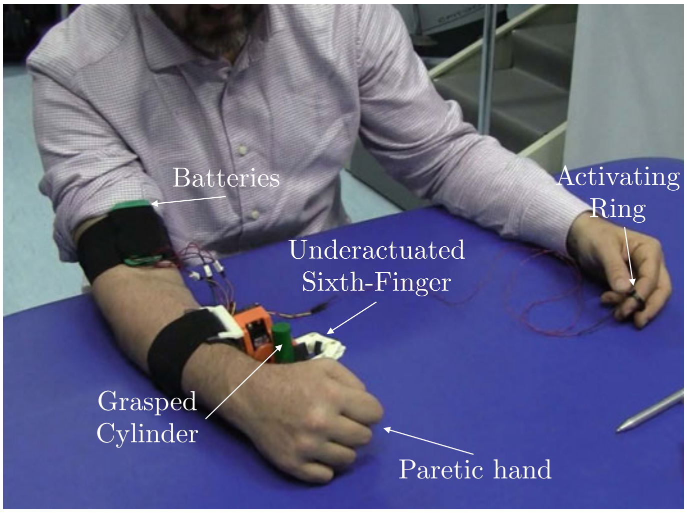


Fig. 8 Grasp a cylinder (12 mm diameter, 5 cm long) using the underactuated compliant version of the Sixth Finger

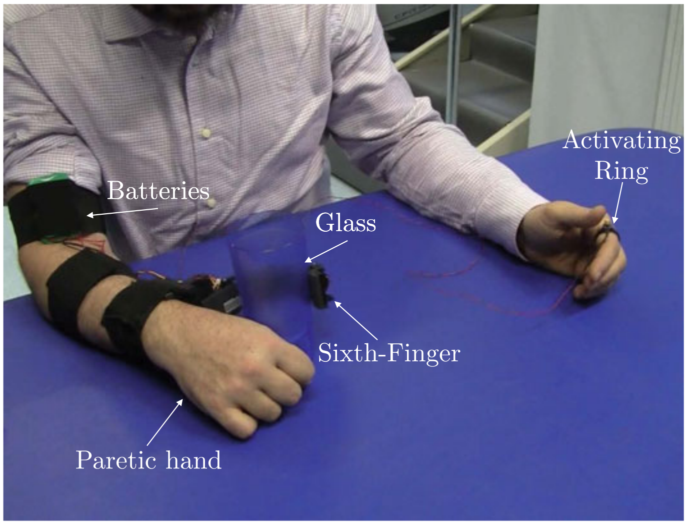


Fig. 9 Setup of the experiment. The subject wears the Sixth Finger device on the paretic limb and activates its motion through a switch placed on a ring worn on the index of the other hand. Addition batteries are used to guarantee complete portability of the device

Table 2 Results of the Frenchay Arm Test for the first patient with and without using the Sixth Finger versions

TASK	No finger	Fully-actuated	Under-actuated	Double finger
Stabilize a ruler	1	1	1	1
Grasp a cylinder	0	1	1	1
Pick up a glass	0	1	1	1
Remove a sprung	0	0	0	0
Comb hair	0	0	0	0

Table 3 Results of the Franchay Arm Test for the second patient with and without using the Sixth Finger versions

TASK	No finger	Fully-actuated	Under-actuated	Double finger
Stabilize a ruler	1	1	1	1
Grasp a cylinder	0	1	1	1
Pick up a glass	0	1	1	1
Remove a sprung	0	0	0	0
Comb hair	0	0	0	0

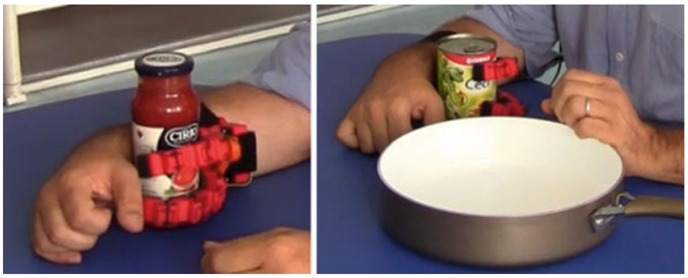


Fig. 10 The Double Sixth Finger as compensatory tool for bimanual tasks. On the *left*, the paretic hand and the device restrain the motion of a tomato jar so the user can unscrew its cap using the healthy hand. On the *right*, the device works together with the paretic hand so to let the patient open a can of beans

used to open it (e.g., unscrewing the cap). Chronic patients with an advanced motor deficit are usually trained to use special tools and techniques so to let them open some of the commercial cans using only one hand. However, this solution limits the possibility of the patients to perform common activities outside their structured kitchens and also reduce the number of products they can use. With the help of the Robotic Sixth Finger the patients can grasp the cans/jars using their paretic limb and then use the healthy hand to open them. In Fig. 10 an example with two different objects is reported.

4 Conclusion

In this paper, we presented the preliminary results concerning the use of a robotic extra-finger for compensation of hand grasping function in patients with a paretic limb. We developed three, one fully actuated and two underactuated, prototypes using rapid prototyping techniques. We believe that at this stage of the research it is useful to explore different solutions in terms of actuation and kinematic structures. Results of the pilot tests showed that all the versions were able to successfully fulfill three out of five tasks in the Frenchay Arm Test in both the patients. The fully actuated finger can also be controlled so to perform different flexion trajectories. This could be useful to achieve different types of grasps according to the task the patient has to perform. Finally, the Double Robotic Sixth Finger showed the highest performance when used in ADL. The two fingers guaranteed a higher grasp stability while the patients were unscrewing the jar's caps. However the encumbrance of the device reduces its wearability and portability.

In general, passive compliance resulted to be more robust and suitable. In future work, we will push in this direction and explore the possible designs of "soft" fingers based on such principles. We are currently testing our devices involving a greater number of subjects so to collect also interesting insights for the extra-finger development. We are also investigating the possibility of using our robotic extra-finger in patients affected by other neurological diseases possibly affecting hand grasping, such as Multiple Sclerosis or Amyotrophic Lateral Sclerosis.

One of the limitations of this approach is the fine manipulation of objects. This feature was out of the main purposes of the use of the Sixth Finger described here. In fact, the extra-finger and patient limb work jointly as the two parts of a one DoF gripper.

Although grasping objects with the paretic limb, without any specific training phase, could already represent a great improvement in everyday life of chronic stroke patients, we are investigating whether the Robotic Sixth Finger can be used also in more complex manipulation tasks.

Acknowledgements The authors are grateful to Dr. David Cioncoloni, Sabrina Taddei and Prof. Simone Rossi from the Dipartimento di Scienze Mediche, Chirurgiche e Neuroscienze, UOC Neurologia e Neurofisiologia Clinica, Brain Investigation & Neuromodulation Lab. (Si-BIN), Siena, Italy, for their suggestions and the help in recruiting and assisting the patients during the pilot experiments.

References

1. Mohammed, S., Amirat, Y.: Towards intelligent lower limb wearable robots: challenges and perspectives - state of the art. In: IEEE International Conference on Robotics and Biomimetics, 2008. ROBIO 2008, pp. 312–317 (2009)

2. Arata, J., Ohmoto, K., Gassert, R., Lambercy, O., Fujimoto, H., Wada, I.: A new hand exoskeleton device for rehabilitation using a three-layered sliding spring mechanism. In 2013 IEEE international conference on robotics and automation (ICRA), pp. 3902–3907 (2013)
3. Davenport, C., Parietti, F., Asada, H.H.: Design and biomechanical analysis of supernumerary robotic limbs. In: ASME 2012 5th annual dynamic systems and control conference joint with the JSME 2012 11th motion and vibration conference, pp. 787–793 (2012)
4. Parietti, F., Asada, H.H.: Dynamic analysis and state estimation for wearable robotic limbs subject to human-induced disturbances. In: IEEE international conference on robotics and automation (ICRA), pp. 3880–3887. IEEE (2013)
5. Atassi, O.A.: Design of a robotic finger for grasping enhancement. Master's thesis, Università degli Studi di Siena (2011)
6. Wu, F., Asada, H.: Bio-artificial synergies for grasp posture control of supernumerary robotic fingers. In: Proceedings of Robotics: Science and Systems, Berkeley, USA (2014)
7. Prattichizzo, D., Malvezzi, M., Hussain, I., Salvietti, G.: The sixth-finger: a modular extra-finger to enhance human hand capabilities. In: Proceedings of the IEEE International Symposium in Robot and Human Interactive Communication, Edinburgh, United Kingdom (2014)
8. Prattichizzo, D., Salvietti, G., Chinello, F., Malvezzi, M.: An object-based mapping algorithm to control wearable robotic extra-fingers. In: Proceedings of the IEEE/ASME International Conference on Advanced Intelligent Mechatronics, Besançon, France (2014)
9. Wu, F.Y., Asada, H.H.: "Hold-and-manipulate" with a single hand being assisted by wearable extra fingers. In: 2015 IEEE International Conference on Robotics and Automation (ICRA), pp. 6205–6212. IEEE (2015)
10. Hussain, I., Salvietti, G., Meli, L., Pacchierotti, C., Prattichizzo, D.: Using the robotic sixth finger and vibrotactile feedback for grasp compensation in chronic stroke patients. In: Proceedings of the IEEE/RAS-EMBS International Conference on Rehabilitation Robotics (ICORR),vol. 0 (Finalist for the Best Student Paper Award. Singapore, Republic of Singapore) (2015)
11. Lum, P.S., Godfrey, S.B., Brokaw, E.B., Holley, R.J., Nichols, D.: Robotic approaches for rehabilitation of hand function after stroke. Am. J. Phys. Med. Rehabil. **91**(11), S242–S254 (2012)
12. Heo, P., Gu, G.M., Lee, S.-J., Rhee, K., Kim, J.: Current hand exoskeleton technologies for rehabilitation and assistive engineering. Int. J. Precis. Eng. Manuf. **13**(5), 807–824 (2012)
13. Michaelsen, S.M., Jacobs, S., Roby-Brami, A., Levin, M.F.: Compensation for distal impairments of grasping in adults with hemiparesis. Exp. Brain Res. **157**(2), 162–173 (2004)
14. Hussain, I., Meli, L., Pacchierotti, C., Salvietti, G., Prattichizzo, D.: Vibrotactile haptic fedback for intuitive control of robotic extra fingers. In: Proceeding of the IEEE World Haptics Conference (WHC), vol. 0 (Chicago, IL) (2015)
15. Heller, A., Wade, D., Wood, V.A., Sunderland, A., Hewer, R.L., Ward, E.: Arm function after stroke: measurement and recovery over the first three months. J. Neurol. Neurosurg. Psychiatry **50**(6), 714–719 (1987)
16. Arduino: Arduino uno, an open-source electronics prototyping platform. http://arduino.cc/
17. Sciavicco, L., Siciliano, B.: Modelling and Control of Robot Manipulators. Springer, Berlin (2000)
18. Schwarz, M., Behnke, S.: Compliant robot behavior using servo actuator models identified by iterative learning control. In: RoboCup 2013: Robot World Cup XVII, pp. 207–218. Springer, Berlin (2014)
19. Dollar, A., Howe, R.D.: Joint coupling and actuation design of underactuated hands for unstructured environments. Mechanics of Structures and Machines, vol. tocheck, p. tocheck (2007) (check this reference and complete the missing data)
20. Carrozza, M.C., Suppo, C., Sebastiani, F., Massa, B., Vecchi, F., Lazzarini, R., Cutkosky, M.R., Dario, P.: The spring hand: development of a self-adaptive prosthesis for restoring natural grasping. Auton. Robots **16**(2), 125–141 (2004)
21. Spong, M.W.: Underactuated mechanical systems. In: Control Problems in Robotics and Automation, pp. 135–150. Springer, Berlin (1998)

22. Birglen, L., Gosselin, C.M.: Kinetostatic analysis of underactuated fingers. IEEE Trans. Robot. Autom. **20**(2), 211–221 (2004)
23. Biagiotti, L., Lotti, F., Melchiorri, C., Vassura, G.: Mechatronic design of innovative fingers for anthropomorphic robot hands. In: 2003 IEEE International Conference on Robotics and Automation (ICRA), vol. 3, pp. 3187–3192 (2003)
24. Brott, T., Adams, H., Olinger, C.P., Marler, J.R., Barsan, W.G., Biller, J., Spilker, J., Holleran, R., Eberle, R., Hertzberg, V.: Measurements of acute cerebral infarction: a clinical examination scale. Stroke **20**(7), 864–870 (1989)

Part IV
Multi-robot Systems

Towards Cooperative Multi-robot Belief Space Planning in Unknown Environments

Vadim Indelman

1 Introduction

Autonomous operation under uncertainty is essential in numerous problem domains, including autonomous navigation, object manipulation, multi-robot localization and tracking, and robotic surgery. As the robot state is never accurately known due to motion uncertainty and imperfect state estimation obtained from partial and noisy sensor measurements, planning future actions should be performed in the belief space - a probability distribution function (pdf) over robot states and additional states of interest.

Belief space planning has been investigated extensively in the last two decades. While the corresponding problem can be described in the framework of partially observable Markov decision process (POMDP), which is known to be computationally intractable for all but the smallest problems [17], several approaches that tradeoff optimal performance with computational complexity have been recently developed. These approaches can be segmented into several categories: point-based value iteration methods, simulation based approaches, sampling based approaches and direct trajectory optimization approaches.

Point-based value iteration methods (e.g. [14, 19]) select a number of representative belief points and calculate a control policy over belief space by iteratively applying value updates to these points. *Simulation-based approaches* (e.g. [23, 24]) generate a few potential plans and select the best policy according to a given metric. They are referred to as simulation-based approaches, since they simulate the evolution of the belief for each potential plan to quantify its quality.

Sampling based approaches (e.g. [1, 6, 21]) discretize the state space using randomized exploration strategies to explore the belief space in search of an optimal plan. While many of these approaches, including probabilistic roadmap (PRM) [13],

V. Indelman (✉)
Department of Aerospace Engineering, Technion - Israel Institute of Technology, 32000 Haifa, Israel
e-mail: vadim.indelman@technion.ac.il

A. Bicchi and W. Burgard (eds.), *Robotics Research*, Springer Proceedings in Advanced Robotics 2, DOI 10.1007/978-3-319-51532-8_27

rapidly exploring random trees (RRT) [15] and RRT* [12], assume perfect knowledge of the state, deterministic control and a known environment, efforts have been devoted in recent years to alleviate these restricting assumptions. These include, for example, the belief roadmap (BRM) [21] and the rapidly-exploring random belief trees (RRBT) [1], where planning is performed in the belief space, thereby incorporating the predicted uncertainties of future position estimates. We note that similar strategies are used to address also informative planning problems (see, e.g., [6]).

Direct trajectory optimization methods (including [9, 18, 20, 25]) calculate locally optimal trajectories and control policies, starting from a given nominal path. Approaches in this category perform planning over a continuous state and action spaces, which is often considered more natural as the robot states (e.g., poses) and controls (e.g., steering angles) are not constrained to few discrete values. For example, Platt et al. [20] apply linear quadratic regulation (LQR) to compute locally optimal policies, while Van den Berg et al. [25] develop a related method using optimization in the belief space and avoiding assuming maximum likelihood observations in predicting the belief evolution. These approaches reduce computational complexity to polynomial at the cost of guaranteeing only locally optimal solutions.

While typically, belief space planning approaches consider the environment is known, in certain scenarios of interest (e.g. navigation in unknown environments) this is not a feasible assumption. In these cases, the environment is either a priori unknown, uncertain or changes dynamically, and therefore should be appropriately modeled as part of the inference and decision making processes. Such a concept was recently developed in [8, 9], where random variables representing the observed environment have been incorporated into the belief and locally optimal motion plans were calculated using a direct trajectory optimization approach. In [7], the approach was extended to a multi-robot belief space planning centralized framework and was used to facilitate active collaborative estimation in unknown environments. Simulation- and sampling-based approaches that consider a priori unknown environments have also been recently developed in the context of active SLAM (see, e.g. [4, 24]). A limitation of these approaches is that the belief only considers the environment observed by planning time and does not reason, in the context of uncertainty reduction, about new environments to be observed in the future as the robot continues exploration.

In this work we alleviate this limitation, considering the problem of cooperative multi-robot autonomous navigation in unknown environments. While it is well known that collaboration between robots can significantly improve estimation accuracy, existing approaches (e.g. [3, 10, 22]) typically focus on the inference part, considering robot actions to be determined externally. On the other hand, active multi-robot SLAM approaches (e.g. [2]) typically focus on coordination aspects and on the trade-off between exploring new regions and reducing uncertainty by re-observing previously mapped areas (performing loop closures). In contrast, in this paper we consider the question - how should the robots act to collaboratively improve state estimation while autonomously navigating to individual goals and operating in unknown environments?

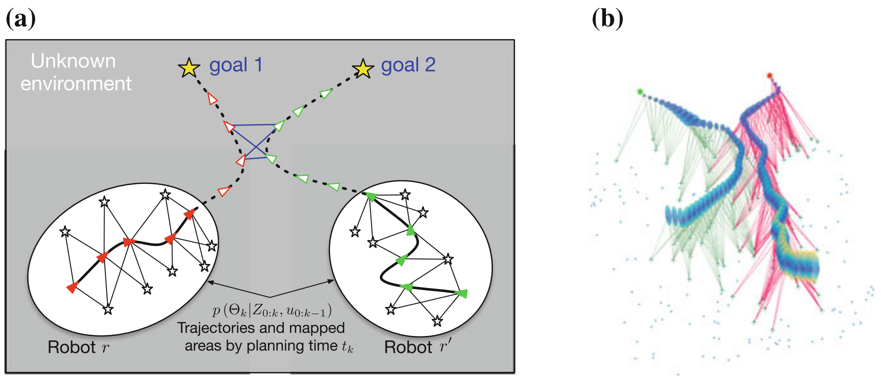

(a)
Unknown environment
goal 1
goal 2
$p(\Theta_k|Z_{0:k}, u_{0:k-1})$
Trajectories and mapped areas by planning time t_k
Robot r
Robot r'
(b)

Fig. 1 **a** Illustration of the proposed concept. Multi-robot indirect constraints representing mutual future observations of unknown environments are shown in *blue*. **b** 3D view of the scenario from Fig. 4b: Robots operate in an unknown environment and follow paths generated by PRM that have been identified by the proposed approach to provide the best estimation accuracy upon reaching the goals. One can observe the mutually-observed 3D points that induce indirect multi-robot constraints involving different time instances; these constraints have been accounted for in the planning phase. Robot initial positions are denoted by $\star$ marks (at the *top* of the figure); uncertainty covariances of robot poses are represented by ellipsoids

Addressing this question requires incorporating multi-robot collaboration aspects into belief space planning. To that end, we present an approach to evaluate the probability distributions of multiple robot states while modeling future observations of mutual areas that are unknown at planning time (Fig. 1a). The key idea is that although the environment may be unknown a priori, or has not been mapped yet, it is still possible to reason in terms of robot actions that will result in the same unknown environments to be observed by multiple robots, possibly at different future time instances. Such observations can be used to formulate non-linear constraints between appropriate robot future states. Importantly, these constraints allow collaborative state estimation without the need for the robots to actually meet each other, in contrast to the commonly used direct relative pose observations that require rendezvous between robots (e.g. [22]). We show how such constraints can be incorporated within a multi-robot belief, given candidate paths that can be generated by any motion planning method. One can then identify the best path with respect to a user-defined objective function (e.g. reaching a goal with minimum uncertainty), and also refine best alternatives using direct trajectory optimization techniques (e.g. [9, 18, 25]).

2 Notations and Problem Formulation

Let x_i^r represent the pose of robot r at time t_i and denote by L_i^r the perceived environment by that robot, e.g. represented by 3D points, by that time. We let Z_i^r represent the local observations of robot r at time t_i, i.e. measurements acquired by its onboard sensors, and define the joint state Θ^r over robot past and current poses and observed 3D points as

$$\Theta_k^r \doteq X_k^r \cup L_k^r, \ X_k^r \doteq \left\{x_0^r, \ldots, x_k^r\right\}. \tag{1}$$

The joint probability distribution function (pdf) over this joint state given local observations $Z_{0:k}^r \doteq \left\{Z_0^r, \ldots, Z_k^r\right\}$ and controls $u_{0:k-1}^r \doteq \left\{u_0^r, \ldots, u_{k-1}^r\right\}$ is given by

$$p\left(\Theta_k^r | Z_{0:k}^r, u_{0:k-1}^r\right) \propto \ p\left(x_0^r\right) \prod_{i=1}^{k} \left[p\left(x_i^r | x_{i-1}^r, u_{i-1}^r\right) p\left(Z_i^r | \Theta_i^{ro}\right)\right], \tag{2}$$

where $\Theta_i^{ro} \subseteq \Theta_i^r$ are the involved random variables in the measurement likelihood term $p\left(Z_i^r | \Theta_i^{ro}\right)$, which can be further expanded in terms of individual measurements $z_{i,j}^r \in Z_i^r$ representing observations of 3D points l_j: $p\left(Z_i^r | \Theta_i^{ro}\right) = \prod_j p\left(z_{i,j}^r | x_i^r, l_j\right)$. The motion and observation models in Eq. (2) are assumed to be with additive Gaussian noise,

$$x_{i+1}^r = f\left(x_i^r, u_i^r\right) + w_i^r, \ \ z_{i,j}^r = h\left(x_i^r, l_j\right) + v_i^r \tag{3}$$

where $w_i \sim N\left(0, \Sigma_w^r\right)$, $v_i \sim N\left(0, \Sigma_v^r\right)$, with Σ_w^r and Σ_v^r representing the process and measurement noise covariance matrices, respectively.

We consider now a group of R collaborating robots, and denote by Θ_k the corresponding joint state

$$\Theta_k \doteq X_k \cup L_k, \ X_k \doteq \left\{X_k^r\right\}_{r=1}^R \tag{4}$$

comprising the past and current poses X_k of all robots, and where L_k represents the perceived environment by the entire group. Assuming a common reference frame between the robots is established, L_k includes all the 3D points in L_k^r for each r, expressed in that reference frame.

The joint pdf over Θ_k, the *belief* at planning time t_k, can now be written as

$$b\left(\Theta_k\right) \ \doteq p\left(\Theta_k | Z_{0:k}, u_{0:k-1}\right) \propto \prod_{r=1}^{R} p\left(\Theta_k^r | Z_{0:k}^r, u_{0:k-1}^r\right), \tag{5}$$

where $u_{0:k-1}$ represents the controls of all robots and is defined as $u_{0:k-1} \doteq \left\{u_{0:k-1}^r\right\}_{r=1}^R$.

The joint belief at a future time t_{k+l} can now be similarly defined as

$$b\left(\Theta_{k+l}\right) \doteq p\left(\Theta_{k+l} | Z_{0:k+l}, u_{0:k+l-1}\right), \tag{6}$$

where $u_{k:k+l-1}$ are future actions for a planning horizon of l steps and $Z_{k+1:k+l}$ are the corresponding observations to be obtained. We will discuss in detail how such a belief can be formulated in the sequel (Sects. 3.1 and 3.2).

We can now define a general multi-robot objective function

$$J\left(u_{k:k+L-1}\right) \doteq \mathbb{E}\left[\sum_{l=0}^{L} c_l\left(b\left(\Theta_{k+l}\right), u_{k+l}\right) + c_L\left(b\left(\Theta_{k+L}\right)\right)\right], \tag{7}$$

that involves L future steps for all robots, and where c_l is the immediate cost function for the lth step. The expectation operator accounts for all the possible future observations $Z_{k+1:k+l}$. While for notational convenience the same number L of future steps is assumed for all robots in Eq. (7), this assumption can be easily relaxed.

Our objective is to find the optimal controls $u^{\star}_{k:k+L-1}$ for all R robots:

$$u^{\star}_{k:k+L-1} = \underset{u_{k:k+L-1}}{\arg\min}\, J\left(u_{k:k+L-1}\right). \tag{8}$$

3 Approach

In this work we show how to incorporate into belief space planning multi-robot collaboration aspects such that estimation accuracy is significantly improved while operating in unknown environments. Our approach extends the state of the art by incorporating into the belief (6) multi-robot constraints induced by multiple robots observing, possibly at different *future* time instances, environments that are *unknown* at planning time. In lack of sources of absolute information (such as reliable GPS, beacons, and known 3D points), these constraints are the *key* for collaboratively improving estimation accuracy.

One can then identify best robot actions or motion plans, according to Eq. (8), among those generated by existing motion planning approaches (e.g. sampling based approaches), or resort to direct optimization techniques to obtain locally optimal solutions in a timely manner. In this work, we focus on the former case, and consider we are given candidate paths for different robots (generated, e.g. by PRM or RRT). A schematic illustration of the proposed approach is shown in Fig. 1.

We start with a recursive formulation of the multi-robot belief (Sect. 3.1) and then discuss in Sect. 3.2 our approach to incorporate into the multi-robot belief future constraints that correspond to mutual observations of unknown scenes. Evaluating the objective function (7) involves simulating belief evolution along candidate robots paths.

3.1 Recursive Formulation of a Multi-robot Belief

We begin with a recursive formulation of the multi-robot belief (6), considering future controls $u_{0:k+l-1}$ for all robots to be given. These are determined from candidate robot paths that are being evaluated, or alternatively in the case of direct trajectory optimization approaches, the controls are determined from either nominal or perturbed robot paths (see, e.g. [9] for further details).

Given future controls for all robots, the multi-robot belief $b\left(\Theta_{k+l}\right)$ at the lth future step can be written recursively as follows (see also Eq. (2)):

$$\begin{aligned} b\left(\Theta_{k+l}\right) &\doteq p\left(\Theta_{k+l}|Z_{0:k+l}, u_{0:k+l-1}\right) \\ &= \eta b\left(\Theta_{k+l-1}\right)\prod_{r=1}^{R} p\left(x^r_{k+l}|x^r_{k+l-1}, u^r_{k+l-1}\right) p\left(Z^r_{k+l}|\Theta^{ro}_{k+l}\right), \end{aligned} \tag{9}$$

where η is a normalization constant, and $p\left(x^r_{k+l}|x^r_{k+l-1}, u^r_{k+l-1}\right)$ and $p\left(Z^r_{k+l}|\Theta^{ro}_{k+l}\right)$ are respectively the motion model and measurement likelihood terms.

We now focus on the measurement likelihood term $p\left(Z^r_{k+l}|\Theta^{ro}_{k+l}\right)$, noting that it appears recursively in Eq. (9), for each look ahead step. As earlier, this term represents sensor observations of the environment (represented e.g. by 3D points), see Eq. (2). However, now, these are future observations of the environment to be made according to robot r's planned motion. It therefore makes sense to distinguish between the following two cases: (a) observation of 3D points from $L_k \subset \Theta_k$ representing environments already mapped by planning time t_k, and (b) observation of new areas that were not previously explored by any of the robots.

The former case allows to plan single- and multi-robot loop closures (e.g. as in [9]), i.e. to quantify the expected information gain due to re-observation of previously mapped areas by any of the robots.

We focus on the latter case, which has not been investigated, to the best of our knowledge, in the context of collaborative active state estimation and uncertainty reduction. Since environments that are unknown at planning time t_k are considered, the key question is how to quantify the corresponding measurement likelihood term.

3.2 Incorporating Future Multi-robot Constraints

Despite the fact that the environments (or objects) to be observed are unknown at planning time, it is still possible to reason in terms of mutual observations of these unknown environments to be made by different robots, possibly at different future time instances. We can then formulate constraints relating appropriate robot states while marginalizing out the corresponding random variables representing the unknown environments.

More specifically, let us consider robots r and r' mutually observing at future times t_{k+l} and t_{k+j}, respectively, an unknown environment represented, e.g., by 3D points $L^{r,r'}_{k+l,k+j}$, with $1 \leq j \leq l$. The joint pdf involving the corresponding states and these 3D points can be written as

$$p\left(x^r_{k+l}, x^{r'}_{k+j}, L^{r,r'}_{k+l,k+j} | z^r_{k+l}, z^{r'}_{k+j}\right) \propto p\left(z^r_{k+l} | x^r_{k+l}, L^{r,r'}_{k+l,k+j}\right) p\left(z^{r'}_{k+j} | x^{r'}_{k+j}, L^{r,r'}_{k+l,k+j}\right)$$

We can now marginalize out the unknown 3D points $L^{r,r'}_{k+l,k+j}$ to get

$$p\left(z^r_{k+l}, z^{r'}_{k+j} | x^r_{k+l}, x^{r'}_{k+j}\right) \propto p\left(x^r_{k+l}, x^{r'}_{k+j} | z^r_{k+l}, z^{r'}_{k+j}\right) = \quad (10)$$

$$= \int p\left(x^r_{k+l}, x^{r'}_{k+j}, L^{r,r'}_{k+l,k+j} | z^r_{k+l}, z^{r'}_{k+j}\right) dL^{r,r'}_{k+l,k+j}, \quad (11)$$

which corresponds to a multi-robot constraint involving different time instances.

In the passive problem setting, i.e. controls and measurements are given, this constraint is typically a nonlinear function that involves the robot poses, say x^r_i and $x^{r'}_j$, and the measured constraint $z^{r,r'}_{i,j}$ which is obtained by matching the measurements z^r_i and $z^{r'}_j$. Typical examples include matching laser scans or images using standard techniques (e.g. ICP, vision-based motion estimation). The corresponding measurement likelihood term can thus be written as

$$p(z^{r,r'}_{i,j} | x^r_i, x^{r'}_j) \propto \exp\left(-\frac{1}{2}\|z^{r,r'}_{i,j} - g\left(x^r_i, x^{r'}_j\right)\|^2_{\Sigma^{MR}_v}\right) \quad (12)$$

where Σ^{MR}_v is the corresponding measurement noise covariance matrix, and g is an appropriate measurement function. For example, this function could represent a nonlinear relative pose constraint.

Coming back to Eq. (10), while in our case the *future* observations are *not* given, the reasoning is very similar: we can denote by $z^{r,r'}_{k+l,k+j}$ the measured constraint that would be obtained by matching z^r_{k+l} and $z^{r'}_{k+j}$ if these were known, and considering, as before, the match is successful (i.e. not outlier), it is possible to quantify the measurement likelihood (10) as

$$p\left(z^{r,r'}_{k+l,k+j} | x^r_{k+l}, x^{r'}_{k+j}\right) \propto \exp\left(-\frac{1}{2}\|z^{r,r'}_{k+l,k+j} - g\left(x^r_{k+l}, x^{r'}_{k+j}\right)\|^2_{\Sigma^{MR}_v}\right) \quad (13)$$

Note the above assumes robots r and r' will observe the same unknown scene from future states x^r_{k+l} and $x^{r'}_{k+j}$. How to determine if two future measurements (e.g. images, laser scans), to be captured from robot poses x^r_{k+l} and $x^{r'}_{k+j}$, will be overlapping, i.e. represent a mutually observed a scene? The answer to this question is scenario specific. For example, in an aerial scenario with robots equipped with downward looking cameras, it is possible to assess if the images are overlapping given robot poses and a rough estimate of height above ground. Ground scenarios allow similar reasoning, however here it is more likely that the same (unknown) scene is observed from multiple views (e.g. autonomous driving with a forward looking camera), and moreover, obstacles, that are unknown at planning time, may prevent two adjacent views to observe a mutual scene in practice.

In this paper we assume one is able to predict if two future poses will mutually observe a scene. Specifically, in Sect. 4 we consider aerial robots with downward facing cameras and take a simplified approach, considering two future poses x^r_{k+l} and $x^{r'}_{k+j}$ to overlap if they are "sufficiently" nearby, quantified by a relative distance below a threshold d. Naturally, more advanced approaches can be considered (e.g. account also for viewpoint variation) and be encapsulated by an indicator function as in [16] - we leave the investigation of these aspects to future research.

Given candidate robot paths it is possible to determine using the above method which future views (poses) will overlap and formulate the corresponding multi-robot constraints (13). In particular, multi-robot constraints between robot r at time t_{k+l} and other robots r' at time t_{k+j} with $0 \leq j \leq l$ can be enumerated as

$$\prod_j p\left(z^{r,r'}_{k+l,k+j}|x^r_{k+l}, x^{r'}_{k+j}\right). \tag{14}$$

We can now write the measurement likelihood term $p\left(Z^r_{k+l}|\Theta^{ro}_{k+l}\right)$ from Eq. (9) as:

$$p\left(Z^r_{k+l}|\Theta^{ro}_{k+l}\right) = \prod_{l_j \in \Theta^{ro}_{k+l}} p\left(z^r_{k+l,j}|x^r_{k+l}, l_j\right) p\left(z^r_{k+l,k+l-1}|x^r_{k+l}, x^r_{k+l-1}\right) \cdot \\ \cdot \prod_j p\left(z^{r,r'}_{k+l,k+j}|x^r_{k+l}, x^{r'}_{k+j}\right). \tag{15}$$

The first product represents observations of previously mapped 3D points $l_j \in L_k$, with Θ^{ro}_{k+l} including those 3D points that are actually visible from x^r_{k+l}. The second term $p\left(z^r_{k+l,k+l-1}|x^r_{k+l}, x^r_{k+l-1}\right)$ denotes a constraint stemming from robot r observing a mutual unknown scene from adjacent views, while the last product represents multi-robot constraints (14) that correspond to different robots observing common areas that have not yet been mapped by planning time t_k. See schematic illustration in Fig. 1a, where these future constraints are shown in blue.

Substituting Eq. (15) into Eq. (9) yields the final expression for $b\left(\Theta_{k+l}\right)$:

$$b\left(\Theta_{k+l}\right) = \eta b\left(\Theta_{k+l-1}\right) \prod_{r=1}^{R} \left[p\left(x^r_{k+l}|x^r_{k+l-1}, u^r_{k+l-1}\right) \prod_{l_j \in \Theta^{ro}_{k+l}} p\left(z^r_{k+l,j}|x^r_{k+l}, l_j\right) \right. \\ \left. p\left(z^r_{k+l,k+l-1}|x^r_{k+l}, x^r_{k+l-1}\right) \cdot \prod_j p\left(z^{r,r'}_{k+l,k+j}|x^r_{k+l}, x^{r'}_{k+j}\right) \right]. \tag{16}$$

Several remarks are in order at this point. First, observe that direct multi-robot constraints, where a robot measures its pose relative to another robot, are naturally supported in the above formulation by considering the same (future) time index, i.e. $p\left(z^{r,r'}_{k+l,k+l}|x^r_{k+l}, x^{r'}_{k+l}\right)$. Of course, being able to formulate constraints involving also different future time instances, as in Eq. (16), provides enhanced flexibility since planning rendezvous between robots is no longer required. Second, observe the formulation (14) is an approximation of the underlying joint pdf of *multiple* views X making observations Z of an unknown scene L, since it only considers *pairwise* potentials. More concretely, marginalizing L out, $p(X|Z) = \int p\left(X, L|Z\right) dL$, introduces mutual information between all views in X, i.e. any two views in X become correlated. Thus, a more accurate formulation than (14) would consider all robot poses observing a mutual scene together. Finally, one could also incorporate reasoning regarding (robust) data association, i.e. whether a match $z^{r,r'}_{k+l,k+j}$ from raw measurements (images, laser scans) z^r_{k+l} and $z^{r'}_{k+j}$ is expected to be an inlier, as for example done in [11] for the passive case. These aspects are left to future research.

3.3 Inference Over Multi-robot Belief Given Controls

Having described in detail the formulation of a multi-robot belief $b\left(\Theta_{k+l-1}\right)$ at each future time t_{k+l}, this section focuses on simulating belief evolution over time given robot controls or paths. As discussed in Sect. 3, this calculation is required both for sampling based motion planning and direct trajectory optimization approaches.

Thus, we are interested in evaluating the belief $b\left(\Theta_{k+l}\right)$ from Eq. (16)

$$b\left(\Theta_{k+l}\right) \equiv p\left(\Theta_{k+l}|Z_{0:k+l}, u_{0:k+l-1}\right) = N\left(\Theta^{\star}_{k+l}, I_{k+l}\right). \tag{17}$$

which is required for evaluating the objective function (7). Observe that for conciseness we are using here $I_{k+l} \equiv I_{k+l|k+l}$ and $\Theta^{\star}_{k+l} \equiv \hat{\Theta}_{k+l|k+l}$.

This process involves a maximum a posteriori (MAP) inference

$$\Theta^{\star}_{k+l} = \underset{\Theta_{k+l}}{\arg\max}\, b\left(\Theta_{k+l}\right) = \underset{\Theta_{k+l}}{\arg\min} \left[-\log b\left(\Theta_{k+l}\right)\right], \tag{18}$$

which also determines the corresponding information matrix $I_{k+l} = \Sigma^{-1}_{k+l}$.

To perform this inference, recall the recursive formulation (9) and denote the MAP inference of the belief at a previous time by $b\left(\Theta_{k+l-1}\right) = N\left(\Theta^{\star}_{k+l-1}, I_{k+l-1}\right)$. The belief at time t_{k+l} can therefore be written as

$$\begin{aligned} -\log b\left(\Theta_{k+l}\right) = &\left\|\Theta_{k+l-1} - \Theta^{\star}_{k+l-1}\right\|^2_{\Sigma_{k+l-1}} + \\ &+ \sum_{r=1}^{R} \left[\left\|x^r_{k+l} - f(x^r_{k+l-1}, u^r_{k+l-1})\right\|^2_{\Sigma_Q} - \log p\left(Z^r_{k+l}|\Theta^{ro}_{k+l}\right)\right] \end{aligned} \tag{19}$$

We now focus on the term $-\log p\left(Z^r_{k+l}|\Theta^{ro}_{k+l}\right)$. Recalling the discussion from Sect. 3.2 and Eq. (15), this term can be written as

$$-\log p\left(Z^r_{k+l}|\Theta^{ro}_{k+l}\right) = \sum_{l_j \in \Theta^{ro}_{k+l}} \left\| z^r_{k+l,j} - h(x^r_{k+l}, l_j)\right\|^2_{\Sigma_v} +$$
$$+ \left\| z^r_{k+l,k+l-1} - g(x^r_{k+l}, x^r_{k+l-1})\right\|^2_{\Sigma_v} + \sum_j \left\| z^{r,r'}_{k+l,k+j} - g(x^r_{k+l}, x^{r'}_{k+j})\right\|^2_{\Sigma_v^{MR}}, \tag{20}$$

where the motion and measurement models f and h are defined in Sect. 2, and the nonlinear function g was introduced in Eqs. (12) and (13). We note that while here we consider the measurement noise covariance Σ_v^{MR} to be constant, one could go further and model also accuracy deterioration, e.g. as the relative distance between robot poses increases.

We now proceed with the MAP inference (18), which, if the future observations Z^r_{k+l} were known, could be solved using standard iterative non-linear optimization techniques (e.g. Gauss-Newton and Levenberg-Marquardt): in each iteration the system is linearized, the delta vector $\Delta\Theta_{k+l}$ is recovered and used to update the linearization point, and the process is repeated until convergence.

Let us first describe in more detail this fairly standard approach, considering for a moment the future measurements Z^r_{k+l} are known. The linearization point $\bar{\Theta}_{k+l}$ is discussed first. Recalling that we are to evaluate belief evolution given robot paths, these paths can be considered as the linearization point for robot poses. On the other hand, in the case of direct trajectory optimization approaches, the nominal controls over the planning horizon can be used to generate the corresponding nominal trajectories according to (similar to the single robot case, see, e.g. [9])

$$\bar{x}^r_{k+l} = \begin{cases} f(\bar{x}^r_{k+l-1}, u^r_{k+l-1}), & l > 1 \\ f(\hat{x}^r_k, u^r_k), & l = 1 \end{cases} \tag{21}$$

The linearization point for the landmarks $L_k \subset \Theta_{k+l}$ (see Sect. 2) is taken as their most recent MAP estimate. We first linearize Eq. (19)

$$-\log b\left(\Theta_{k+l}\right) = \left\| B_{k+l}\Delta\Theta_{k+l}\right\|^2_{\Sigma_{k+l-1}} +$$
$$+ \sum_{r=1}^{R}\left[\left\| F^r_{k+l}\Delta\Theta_{k+l} - b^r_{k+l}\right\|^2_{\Sigma_Q} - \log p\left(Z^r_{k+l}|\Theta^{ro}_{k+l}\right)\right] \tag{22}$$

and then linearize the term $-\log p\left(Z^r_{k+l}|\Theta^{ro}_{k+l}\right)$ from Eq. (20):

$$-\log p\left(Z^r_{k+l}|\Theta^{ro}_{k+l}\right) = \sum_{l_j \in \Theta^{ro}_{k+l}} \left\| H^r_{k+l,j}\Delta\Theta_{k+l} - b^r_{k+l,j}\right\|^2_{\Sigma_v} + \tag{23}$$
$$+ \left\| G^r_{k+l,k+l-1}\Delta\Theta_{k+l} - b^r_{k+l,k+l-1})\right\|^2_{\Sigma_v} + \sum_j \left\| G^{r,r'}_{k+l,k+j}\Delta\Theta_{k+l} - b^{r,r'}_{k+l,k+j})\right\|^2_{\Sigma_v^{MR}},$$

where the matrices F, H and G and the vectors b are the appropriate Jacobians and right-hand-side (rhs) vectors. The binary matrix B_{k+l} in Eq. (22) is conveniently defined such that $B_{k+l}\Delta\Theta_{k+l} = \Delta\Theta_{k+l-1}$.

Using the relation $\Sigma^{-1} \equiv \Sigma^{-\frac{T}{2}} \Sigma^{-\frac{1}{2}}$ to switch from $\|a\|^2_\Sigma$ to $\|\Sigma^{-\frac{1}{2}} a\|^2$ and stacking all the Jacobians and rhs vectors into $\mathcal{A}_{k+l}$ and $\breve{b}_{k+l}$, respectively, we get

$$\Delta\Theta^\star_{k+l} = \underset{\Delta\Theta_{k+l}}{\arg\min} \left\| \mathcal{A}_{k+l} \Delta\Theta_{k+l} - \breve{b}_{k+l} \right\|^2 . \tag{24}$$

The a posteriori information matrix I_{k+l} of the joint state vector Θ_{k+l} can thus be calculated as $I_{k+l} = \mathcal{A}^T_{k+l} \mathcal{A}_{k+l}$.

This constitutes the first iteration of the nonlinear optimization. Recalling again that the future observations Z^r_{k+l} are unknown, it is not difficult to show [9] that, while the a posteriori information matrix I_{k+l} is not a function of these observations, the equivalent rhs vector $\breve{b}_{k+l}$ from Eq. (24) does depend on Z^r_{k+l}. This presents difficulties in carrying out additional iterations as the linearization point itself becomes a function of the unknown random variables Z^r_{k+l}.

As common in related works (e.g. [9, 18, 20, 25]), we assume a single iteration sufficiently captures the impact of a candidate action(s). Alternatively, to better predict uncertainty evolution, one could resort to using the unscented transformation, as in [5], or to particle filtering techniques. Furthermore, for simplicity in this paper we also make the maximum-likelihood measurement assumption, according to which a future measurement z is assumed equal to the predicted measurement using the most recent state estimate. As a result, it can be shown that the rhs vector $\breve{b}_{k+l}$ becomes zero and thus $\Theta^\star_{k+l} = \bar{\Theta}_{k+l}$. We note one could avoid making this assumption altogether at the cost of more complicated expressions, see, e.g. [9, 25].

To summarize, the output of the described inference procedure is a Gaussian that models the multi-robot belief as in Eq. (17): $b(\Theta_{k+l}) = N\left(\Theta^\star_{k+l}, I_{k+l}\right)$.

3.4 Evaluation of Candidate Paths

Given candidate paths for robots in the group, one can identify the best candidates by evaluating the objective function J from Eq. (7) for different path combinations. Such a process involves simulating belief evolution along the candidate paths of different robots in the group, as discussed in Sect. 3.3, while accounting for multi-robot collaboration in terms of mutual observations of unknown environments (as discussed in Sect. 3.2).

4 Simulation Results

In this section we demonstrate the proposed approach considering the problem of multi-robot autonomous navigation while operating in unknown GPS-deprived environments. We consider an aerial scenario, where each robot has its own goal and the objective is to reach these goals in minimum time but also with highest accuracy. This can be quantified by the following objective function:

$$J = \sum_{r=1}^{R} \left[\kappa^r t_{goal}^r + (1-\kappa^r) tr \left(\Sigma_{goal}^r \right) \right], \tag{25}$$

where Σ_{goal}^r and t_{goal}^r represent, respectively, the covariance upon reaching the goal and time of travel (or path length) for robot r. The parameter $\kappa^r \in [0, 1]$ weights the importance of each term.

As the environment is unknown and there are no beacons, radio sources or any other means to reset estimation error, the robots can only rely on onboard sensing capabilities and collaboration with each other to reduce drift as much as possible. We assume each robot is equipped with camera and range sensors and can observe natural landmarks in the environment, which are used to estimate robot pose within a standard SLAM framework. However, since the environment is unknown ahead of time, these landmarks are discovered on the fly while the planning process has

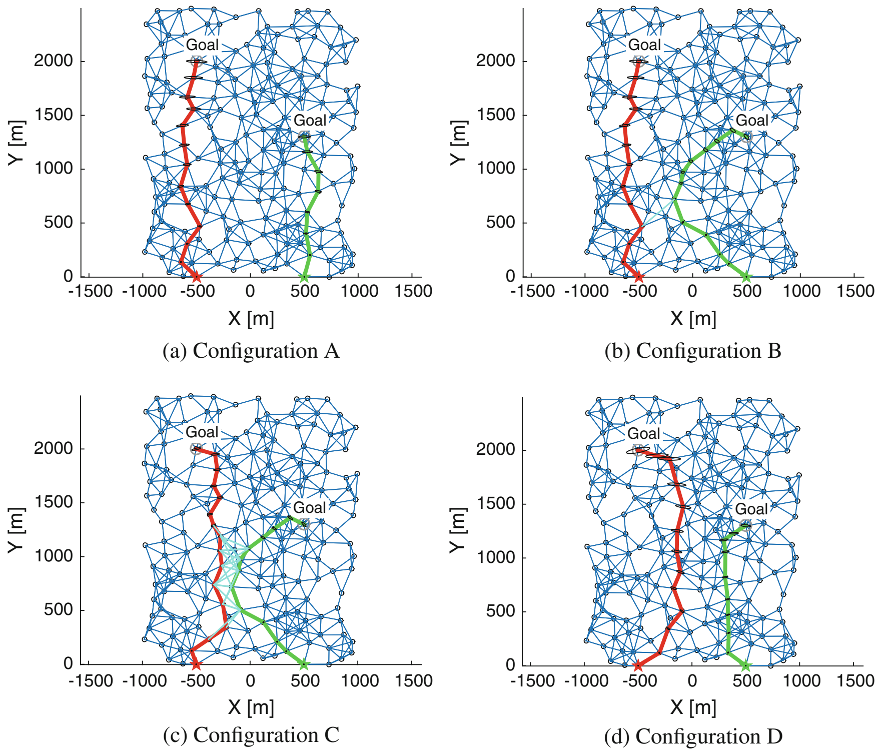
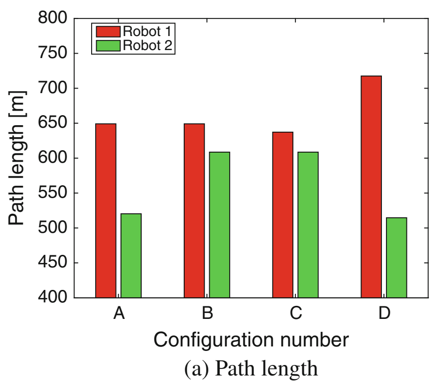
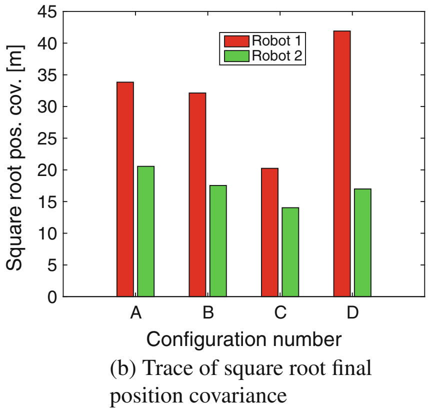


Fig. 2 Different candidate paths for *red* and *green* robots calculated over a PRM. Robot initial positions are denoted by ⋆ marks; each robot has to navigate to a different goal, while operating in an unknown environment. The figures show the covariance evolution along each path. Multi-robot constraints have been incorporated (denoted by *cyan color*) whenever robot poses are sufficiently close, which happens mainly in (**c**); as a result, uncertainty covariances are drastically reduced. Note these constraints involve different *future* time instances. Covariances were artificially inflated by a constant factor for visualization - actual values are shown in Fig. 3



(a) Path length



(b) Trace of square root final position covariance

Fig. 3 Quantitative comparison between the four alternatives shown in Fig. 2: **a** Path length; **b** covariance upon reaching the goals. Multi-robot constraints lead to lowest predicted uncertainty represented by Configuration C from Fig. 2c

access only to environments observed by planning time (Sect. 3). Initial relative poses between the robots are assumed to be known, such that the robots have a common reference frame - approaches that relax this assumption do exist (e.g. [11]).

In this basic study we use a state of the art sampling based motion planning approach, a probabilistic roadmap (PRM) [13], to discretize the environment and generate candidate paths for different robots over the generated roadmap. Figure 2 shows some of these candidate paths considering a scenario of two robots starting operating from different locations. In each case we also show the belief evolution (in terms of uncertainty covariance) along each path, calculated as described in Sect. 3.3, and the multi-robot constraints that have been incorporated into the appropriate beliefs (denoted by cyan color). In the current implementation, these constraints, possibly involving different future time instances, are formulated between any two poses with relative distance closer than d meters. We use $d = 300$ m for this threshold parameter (in the considered scenario the aerial robots height is about 500 m). More advanced methods could be implemented of course, considering also viewpoint variation and incorporating statistical knowledge.

As seen in Fig. 2, only in two of the considered cases (Fig. 2b, c), robot paths were sufficiently close to facilitate multi-robot constraints within belief space planning. In practice, however, only in the latter case numerous informative constraints have been incorporated. Figure 3 compares between the two terms in the considered objective function (25), path length and uncertainty upon reaching the goal, for the candidate paths shown in Fig. 2.

The lowest predicted uncertainty covariances are obtained for candidate paths with identified multi-robot constraints as shown in Fig. 3b. In particular, the predicted uncertainty is reduced by about 40% from 35 m to below 20 m for the first (red) robot. There is a price to pay, however, in terms of path lengths (or time of arrival): as shown in Fig. 3a, to attain these levels of uncertainty, the path of the second (gree) robot is not the shortest among the considered candidate paths. The decision what solution is the best therefore depends on the parameter κ from Eq. (25) that weights the importance of each term in the objective function.

Next, we consider actual performance while navigating to pre-defined goals in unknown environments using as controls the identified robot paths in the planning phase described above. The results are shown in Fig. 3 for two alternatives from Fig. 2a, c. Only the latter included multi-robot constraints within planning. One can observe that also in practice, using controls from Configuration C drives the robots sufficiently close to make mutual observations of 3D points (that were unknown at planning time) and as a result significantly improve estimation accuracy for both robots (see Figs. 4c, d, and 1b for a 3D view).

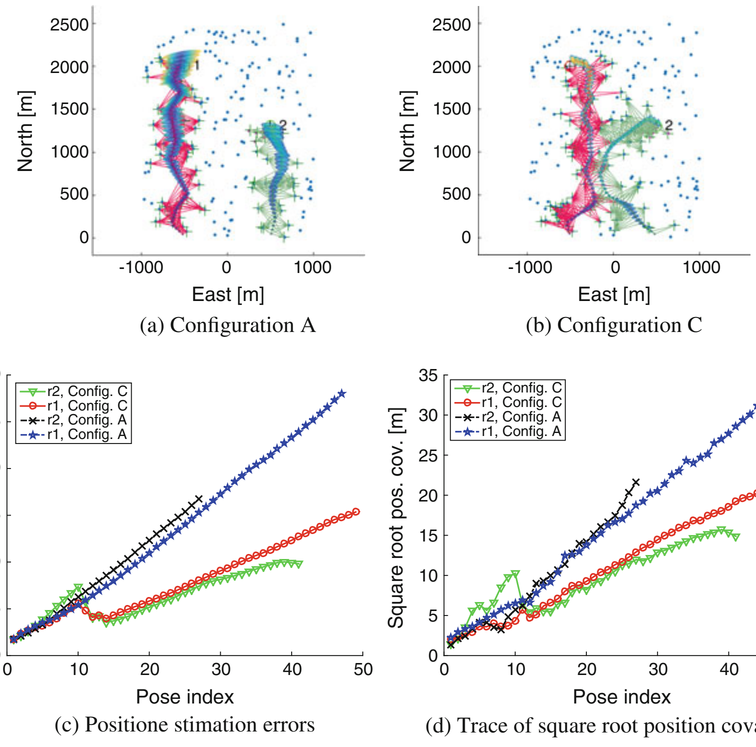


Fig. 4 Autonomous navigation to goals according to identified robot paths in the planning phase. The environment, represented by a sparse set of landmarks, is initially unknown and only gradually discovered. Figures **a** and **b** show robot trajectories and landmark observations using paths defined, respectively, by Configuration **a** and **c** (see Fig. 2). The latter involves numerous mutual observations of landmarks, that induce indirectly multi-robot constraints. A 3D view is also shown in Fig. 1b. Figures **c** and **d** show the corresponding estimation errors and developing covariance over time, which, in overall, agree with the predicted belief evolution from Fig. 3b

5 Discussion and Future Work

Results from the previous section indicate estimation accuracy can be significantly improved by modeling multi-robot mutual observations of unknown areas within belief space planning. More generally, we believe similar reasoning can be used to improve multi-robot collaboration aspects while operating also in uncertain, possibly dynamic, environments.

In this basic study we have made several simplifying assumptions and did not address some of the challenges that are expected to arise in practical applications.

- Obstacles: While initially the environment is unknown, it may be that after some time obstacles are identified as the robots continue in exploration. These obstacles can be efficiently avoided upon discovery by discarding appropriate paths, as commonly done in sampling based approaches.
- Scalability: Although current implementation uses PRM, our approach can be formulated within any motion planning algorithm. The combinatorial problem associated with evaluating candidate trajectories of different robots is a topic of future research. We note approaches addressing related problems have been actively developed in recent years (e.g. [16]). An interesting direction is to also consider generalization of BRM and RRBT to the multi-robot case. A complimentary aspect is to consider direct trajectory optimization approaches, which could allow reducing sampling resolution.
- Belief consistency: While here we consider a centralized approach, decentralized or distributed approaches are often more suitable in practice for numerous reasons. Resorting to these architectures requires the beliefs maintained by different robots to be consistent with each other.

6 Conclusions

We presented an approach for collaborative multi-robot belief space planning while operating in unknown environments. Our approach advances the state of the art in belief space planning by reasoning about observations of environments that are unknown at planning time. The key idea is to incorporate within the belief constraints that represent multi-robot observations of unknown mutual environments. These constraints can involve different future time instances, thereby providing enhanced flexibility to the group as rendezvous are no longer necessary. The corresponding formulation facilitates an active collaborative state estimation framework. Given candidate robot actions or trajectories, it allows to determine best trajectories according to a user-defined objective function, while modeling future multi-robot interaction and its impact on the belief evolution. Candidate robot trajectories can be generated by existing motion planning algorithms, and most promising candidates could be further refined into locally optimal solutions using direct trajectory optimization approaches. The approach was demonstrated in simulation considering the problem of cooperative autonomous navigation in unknown environments, yielding significantly reduced estimation errors.

Acknowledgements This work was partially supported by the Technion Autonomous Systems Program.

References

1. Bry, A., Roy, N.: Rapidly-exploring random belief trees for motion planning under uncertainty. In: IEEE International Conference on Robotics and Automation (ICRA), pp. 723–730 (2011)
2. Burgard, W., Moors, M., Stachniss, C., Schneider, F.: Coordinated Multi-robot Exploration. IEEE Trans. Robot. (2005)
3. Carlone, L., Kaouk Ng, M., Du, J., Bona, B., Indri, M.: Rao-Blackwellized particle filters multi robot SLAM with unknown initial correspondences and limited communication. In: IEEE International Conference on Robotics and Automation (ICRA), pp. 243–249 (2010)
4. Chaves, S.M., Kim, A., Eustice, R.M.: Opportunistic sampling-based planning for active visual slam. In: IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp. 3073–3080. IEEE, New York (2014)
5. He, R., Prentice, S., Roy, N.: Planning in information space for a quadrotor helicopter in a GPS-denied environment. In: IEEE International Conference on Robotics and Automation (ICRA), pp. 1814–1820 (2008)
6. Hollinger, G.A., Sukhatme, G.S.: Sampling-based robotic information gathering algorithms. Int. J. Robot. Res. 1271–1287 (2014)
7. Indelman, V.: Towards multi-robot active collaborative state estimation via belief space planning. In: IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (2015)
8. Indelman, V., Carlone, L., Dellaert, F.: Towards planning in generalized belief space. In: The 16th International Symposium on Robotics Research. Singapore (2013)
9. Indelman, V., Carlone, L., Dellaert, F.: Planning in the continuous domain: a generalized belief space approach for autonomous navigation in unknown environments. Int. J. Robot. Res. **34**(7), 849–882 (2015)
10. Indelman, V., Gurfil, P., Rivlin, E., Rotstein, H.: Distributed vision-aided cooperative localization and navigation based on three-view geometry. Robot. Auton. Syst. **60**(6), 822–840 (2012)
11. Indelman, V., Nelson, E., Michael, N., Dellaert, F.: Multi-robot pose graph localization and data association from unknown initial relative poses via expectation maximization. In: IEEE International Conference on Robotics and Automation (ICRA) (2014)
12. Karaman, S., Frazzoli, E.: Sampling-based algorithms for optimal motion planning. Int. J. Robot. Res. **30**(7), 846–894 (2011)
13. Kavraki, L.E., Svestka, P., Latombe, J.-C., Overmars, M.H.: Probabilistic roadmaps for path planning in high-dimensional configuration spaces. IEEE Trans. Robot. Autom. **12**(4), 566–580 (1996)
14. Kurniawati, H., Hsu, D., Lee, W.S.: Sarsop: Efficient point-based pomdp planning by approximating optimally reachable belief spaces. In: Robotics: Science and Systems (RSS), vol. 2008 (2008)
15. LaValle, S.M., Kuffner, J.J.: Randomized kinodynamic planning. Intl. J. Robot. Res. **20**(5), 378–400 (2001)
16. Levine, D., Luders, B., How, J.P.: Information-theoretic motion planning for constrained sensor networks. J. Aerosp. Inf. Syst. **10**(10), 476–496 (2013)
17. Papadimitriou, C., Tsitsiklis, J.: The complexity of markov decision processes. Math. Oper. Res. **12**(3), 441–450 (1987)

18. Patil, S., Kahn, G., Laskey, M., Schulman, J., Goldberg, K., Abbeel, P.: Scaling up gaussian belief space planning through covariance-free trajectory optimization and automatic differentiation. In: International Workshop on the Algorithmic Foundations of Robotics (2014)
19. Pineau, J., Gordon, G.J., Thrun, S.: Anytime point-based approximations for large pomdps. J. Artif. Intell. Res. **27**, 335–380 (2006)
20. Platt, R., Tedrake, R., Kaelbling, L.P., Lozano-Pérez, T.: Belief space planning assuming maximum likelihood observations. In: Robotics: Science and Systems (RSS), pp. 587–593 (2010)
21. Prentice, S., Roy, N.: The belief roadmap: efficient planning in belief space by factoring the covariance. Int. J. Robot. Res. (2009)
22. Roumeliotis, S.I., Bekey, G.A.: Distributed multi-robot localization. IEEE Trans. Robot. Autom. (2002)
23. Stachniss, C., Grisetti, G., Burgard, W.: Information gain-based exploration using rao-blackwellized particle filters. In: Robotics: Science and Systems (RSS), pp. 65–72 (2005)
24. Valencia, R., Morta, M., Andrade-Cetto, J., Porta, J.M.: Planning reliable paths with pose SLAM. IEEE Trans. Robot. (2013)
25. Van Den Berg, J., Patil, S., Alterovitz, R.: Motion planning under uncertainty using iterative local optimization in belief space. Int. J. Robot. Res. **31**(11), 1263–1278 (2012)

Collision-Free Reactive Mission and Motion Planning for Multi-robot Systems

Jonathan A. DeCastro, Javier Alonso-Mora, Vasumathi Raman, Daniela Rus and Hadas Kress-Gazit

1 Introduction

We aim to synthesize correct-by-construction controllers for a team of robots performing high-level tasks that capture locomotion and actuation. Towards the goal of capable human-robot teams, the tasks we consider are reactive, requiring each robot to react and adapt to changes in the environment (e.g. the motion of other robots or people) at runtime. It has been demonstrated that reactive task specifications written in linear temporal logic (LTL) can be automatically converted into high-level plans that compose basic (atomic) actions to fulfill the task [10]. For example, consider two robots tasked with patrolling the rooms of a house in order to remove garbage and pick up misplaced toys. For high-level synthesis, the atomic actions "remove garbage" and "pick up toys" are assumed to be perfectly executable: they are treated as black boxes in a *discrete abstraction* implicit to the specified task. When a controller is synthesized for this high-level mission specification, it therefore does not govern the design of these low-level actions, nor the behavior of the

Jonathan DeCastro and Javier Alonso-Mora contributed equally to this work

J.A. DeCastro (✉) · H. Kress-Gazit
Cornell University, Ithaca, NY, USA
e-mail: jad455@cornell.edu

H. Kress-Gazit
e-mail: hadaskg@cornell.edu

J. Alonso-Mora (✉) · D. Rus
Massachusetts Institute of Technology, Cambridge, MA, USA
e-mail: jalonsom@mit.edu

D. Rus
e-mail: rus@mit.edu

V. Raman
California Institute of Technology, Pasadena, CA, USA
e-mail: vasu@caltech.edu

A. Bicchi and W. Burgard (eds.), *Robotics Research*, Springer Proceedings in Advanced Robotics 2, DOI 10.1007/978-3-319-51532-8_28

dynamic obstacles – e.g. the inhabitants of the household – that may interfere with their execution. In this work, we address the challenge of ensuring collision-freeness of the generated motion plans, bridging this disconnect between the high-level plan and the low-level actions.

Our approach efficiently abstracts collisions between the robots and dynamic obstacles, and automatically synthesizes a controller for each robot such that the team satisfies the high-level specification. Synthesis is made tractable using a local navigation controller for collision-avoidance, eliminating the need to explicitly model dynamic obstacles in the discrete abstraction. We show that we are able to preserve the global guarantees on task satisfaction using a local method for collision avoidance. This is significant because local planning methods are myopic, and usually do not yield global guarantees in multi-agent settings due to the threat of deadlock (a robot is unable to make forward progress) or livelock (the robot is trapped in an infinite cycle without ever achieving its goals).

Our method applies to the general case of motion planning tasks for multi-robot systems involving unmodeled and uncontrolled dynamic obstacles. We reduce the worst-case conservatism with respect to uncontrolled agents and dynamic obstacles that is typical of most approaches based on reactive synthesis (e.g. [15]). Our results have major implications on the scalability of controller synthesis for dynamic and partially-unmodeled environments.

1.1 Related Work

*High-level Reactive Synthesis.*Reactive synthesis offers a user-friendly approach to the control of complex robotic systems [10], and is especially compelling given the complex nature of multi-agent scenarios. Correct-by-construction reactive controllers have been extended with notions of optimality [15] and distributed teaming [5]. In most approaches, moving obstacles are modeled in a discrete manner as part of the abstraction, leading to over-conservative restrictions like requiring robots to be at least one region apart. Synthesis in dynamic environments thus presents a crucial dilemma: explicitly modeling the state of all other agents is computationally prohibitive, but incomplete models can obliterate the guarantees afforded by the planner. To address the state-explosion problem when modeling uncontrollable agents, [18] formulate an incremental procedure that adapts the number of agents considered in the plan depending on available computational resources. On the other hand, the authors in [12] make local modifications to the synthesized strategy when new elements of the environment are discovered that violate the original assumptions. In contrast to previous work on synthesis for multi-robot tasks, our method preserves guarantees via local collision avoidance, only requiring local awareness of the robot's surroundings. While we also update our specification in a systematic fashion, we do so offline (prior to execution), such that the synthesized strategies preserve guarantees at runtime. Our approach to providing feedback on failed specifications, described in Sect. 4.2, is inspired by recent formal approaches to automated assumption generation [3, 6, 11] and explaining the cause of failure [13].

Collision Avoidance. Online reactive methods, such as [2], typically do not provide global mission fulfillment guarantees. We leverage and extend this work to enforce collision avoidance and motion constraints in the short time horizon, while relying on the high-level planner for guidance to fulfill the global mission.

1.2 Contribution

Our contribution is a holistic synthesis approach that leverages high-level mission planning and low-level motion planning to provably achieve collision-free high-level behaviors in dynamic environments. Local planning capabilities are abstracted in a manner that allows dynamic obstacles to remain unmodeled at the high level during synthesis, and the high level provides deadlock resolution strategies that ensure task satisfaction.

We further contribute:

(a) Automatic feedback-generation for revising specifications. We automatically generate human-comprehensible assumptions in LTL that, if satisfied by the controlled robots and the dynamic obstacles, would ensure correct behavior.
(b) An optimization-based method that extends [2] for synthesizing controllers that guarantee real-time collision avoidance with static and dynamic obstacles in 3D environments while remaining faithful to the robot's dynamics.

Experimental results with ground robots and simulated quadrotors are discussed.

2 Preliminaries

Scalars are denoted by x and vectors by $\mathbf{x} \in \mathbb{R}^n$, with n denoting the dimension of the workspace. The robot's current position is denoted by $\mathbf{p} \in \mathbb{R}^n$ and its current velocity by $\mathbf{v} = \dot{\mathbf{p}}$. A map of the workspace $W \subset \mathbb{R}^n$ is considered, and formed by a set of static obstacles (given by a list of polytopes) $\mathcal{O} \subset \mathbb{R}^n$. For high-level synthesis, the map is abstracted by a set of discrete regions $\mathcal{R} = \{R_1 \dots R_p\}$ covering the map W, where the open sets $R_\alpha \subseteq W$.

2.1 Linear Temporal Logic

LTL formulas are defined over the set AP of atomic (Boolean) propositions by the recursive grammar $\varphi ::= \pi \in AP \mid \varphi_1 \wedge \varphi_2 \mid \neg\varphi \mid \bigcirc\varphi \mid \varphi_1 \, \mathcal{U} \, \varphi_2$. From the Boolean operators $\wedge$ "conjunction" and $\neg$ "negation", and the temporal operators $\bigcirc$ "next" and $\mathcal{U}$ "until", the following operators are derived: "disjunction" $\vee$, "implication" $\Rightarrow$, "equivalence" $\Leftrightarrow$, "always" $\square$, and "eventually" $\diamondsuit$. Refer to [16] for a description of the semantics of LTL. Let AP represent the set of atomic propositions, consisting of *environment* propositions ($\mathcal{X}$) corresponding to thresholded sensor values, and *system* propositions ($\mathcal{Y}$) corresponding to the robot's actions and location with respect to a partitioning of the workspace. The value of each $\pi \in \mathcal{X} \cup \mathcal{Y}$ is the abstracted binary state of a low-level component.

Definition 1 (*Reactive Mission Specification*) A *Reactive Mission Specification* is an LTL formula of the form $\varphi = \varphi_i^e \wedge \varphi_t^e \wedge \varphi_g^e \implies \varphi_i^s \wedge \varphi_t^s \wedge \varphi_g^s$, with s and e standing for 'system' and 'environment', such that

- φ_i^e, φ_i^s are formulas for the *initial conditions* free of temporal operators.
- φ_t^e, φ_t^s are the *safety conditions* (transitions) to be satisfied always, and are of the form $\Box\psi$, where ψ is a Boolean formula over $AP \cup \bigcirc AP$.
- φ_g^e, φ_g^s are the *liveness conditions* (goals) to be satisfied infinitely often, with each taking the form $\Box\Diamond\psi$, with ψ a Boolean formula over $AP \cup \bigcirc AP$.

A strategy automaton that *realizes* a reactive mission specification φ is a deterministic strategy that, given a finite sequence of truth assignments to the variables in $\mathcal{X}$ and $\mathcal{Y}$, and the next truth assignment to variables in $\mathcal{X}$, provides a truth assignment to variables in $\mathcal{Y}$ such that the resulting infinite sequence satisfies φ. If such a strategy can be found, φ is *realizable*. Otherwise, it is *unrealizable*. Strategy automata for φ of the form above can be synthesized [4], and converted into hybrid controllers for robotic systems by invoking atomic controllers [10]. These controllers are *reactive*: they respond to sensor events at runtime.

2.2 LTL Encoding for Multi-robot Tasks

We adopt an LTL encoding that is robust to the inherent variability in the duration of inter-region robot motion in continuous environments [14]. Let $AP_{\mathcal{R}} = \{\pi_\alpha^i \mid R_\alpha \in \mathcal{R}\}$ be the set of Boolean propositions representing the workspace regions, such that $\pi_\alpha^i \in AP_{\mathcal{R}}$ is `True`when robot i is physically in R_α for $\alpha \in [1, p]$. We call π_α^i in $AP_{\mathcal{R}} \subseteq \mathcal{X}$ a *completion* proposition, signaling when robot i reaches R_α. We also define the set $AP_{\mathcal{R}}^{act} \subseteq \mathcal{Y}$ that captures robot commands that *initiate* movement between regions. We call $\pi_{act,\alpha}^i$ in $AP_{\mathcal{R}}^{act}$ an *activation* variable for moving to R_α. Non-motion actions are handled similarly.

Definition 2 (*LTL Encoding of Motion* [14]) A task encoding that admits arbitrary controller execution durations is

$$\psi_t^s: \bigwedge_{\substack{\pi_\alpha^i \in AP_{\mathcal{R}},\\ i \in [1, n_{robots}]}} \Box(\bigcirc\pi_\alpha^i \Rightarrow \bigvee_{R_\beta \in Adj(R_\alpha)} \bigcirc\pi_{act,\beta}^i),$$

$$\psi_t^e: \bigwedge_{\substack{\pi_\alpha^i \in AP_{\mathcal{R}},\\ R_\beta \in Adj(R_\alpha),\\ i \in [1, n_{robots}]}} \Box(\pi_\alpha^i \wedge \pi_{act,\beta}^i \Rightarrow \bigcirc\pi_\alpha^i \vee \bigcirc\pi_\beta^i),$$

$$\psi_g^e: \Box\Diamond \bigwedge_{\substack{i \in [1, n_{robots}]\\ \pi_{act,\alpha}^i \in AP_{\mathcal{R}}^{act} \cup AP_{\mathcal{A}}^{act}}} \left(\left(\pi_{act,\alpha}^i \wedge \bigcirc(\pi_\alpha^i \vee \neg\pi_{act,\alpha}^i)\right) \vee \left(\neg\pi_{act,\alpha}^i \wedge \bigcirc(\neg\pi_\alpha^i \vee \pi_{act,\alpha}^i)\right)\right),$$

where $Adj : \mathcal{R} \to 2^{\mathcal{R}}$ is an adjacency relation on regions in $\mathcal{R}$ and n_{robots} is the number of robots. The ψ_t^s-formula is a system safety condition describing which actions can occur ($\bigcirc\pi^i_{act,\beta}$) given the observed completion variables ($\bigcirc\pi^i_{\alpha}$). Formula ψ_t^e captures the allowed transitions ($\bigcirc\pi^i_{\beta}$) given past completion (π^i_{α}) and activation ($\pi^i_{act,\beta}$) variables. Formula ψ_g^e enforces that every motion and every action eventually completes as long as the activation variable is held fixed. Both ψ_t^e and ψ_g^e are included as conjuncts to the antecedent of φ.

2.3 Local Motion Planning and Robot Dynamics

A collision-free local motion for each robot is computed independently and online based on the current transition in the strategy automaton. We build on the work on distributed Reciprocal Velocity Obstacles with motion constraints [1], and its recent extension to aerial vehicles [2]. Letting $t \in \mathbb{R}_+$ denote time and t_k the current time instant, we define the relative time $\tilde{t} = t - t_k \in [0, \infty)$ and the time horizon of the local planner $\tau > 0$, greater than the required time to stop.

For a robot, a set of candidate local trajectories is considered, each defined by $\mathbf{p}^{robot}(\tilde{t}) = f(\mathbf{z}, \mathbf{u}, \tilde{t})$, continuous in the initial state $\mathbf{z} = [\mathbf{p}, \dot{\mathbf{p}}, \ddot{\mathbf{p}}, \dots]$ of the robot, respecting its dynamic constraints and given by an appropriate controller converging to a straight-line reference trajectory $\mathbf{p}_{\text{ref}}(\tilde{t}) = \mathbf{p} + \mathbf{u}\tilde{t}$ of constant velocity $\mathbf{u} \in \mathbb{R}^n$ and starting at the current position $\mathbf{p}$ of the robot. Local trajectories are now parametrized by $\mathbf{u}$. Suitable controllers include LQR and second order exponential curves, for ground robots [1] and quadrotors [2]. For a given robotic platform and controller, initial state $\mathbf{z}$ and reference velocity $\mathbf{u}$, the maximum deviation (initial position independent) between the reference and the simulated trajectory is

$$\gamma(\mathbf{z}, \mathbf{u}) = \max_{\tilde{t}>0} ||(\mathbf{p} + \tilde{t}\mathbf{u}) - f(\mathbf{z}, \mathbf{u}, \tilde{t})||_2. \tag{1}$$

Maximal errors $\gamma(\mathbf{z}, \mathbf{u})$ are precomputed, and stored for on-line use, for the low-level controller $f(\mathbf{z}, \mathbf{u}_i, \tilde{t})$ and a discretization of initial states $\mathbf{z}$ and reference velocities $\mathbf{u}$.

The idea of the method is as follows: (a) the radius of the robot is enlarged by a value $\varepsilon > 0$ for collision avoidance, computed with respect to the reference trajectories $\mathbf{p} + \mathbf{u}t$ and (b) the local trajectories are limited to those with a tracking error below ε with respect to their reference trajectory. At each time-step an optimal reference velocity $\mathbf{u}^* \in \mathbb{R}^n$ is obtained by solving a convex optimization in $\mathbb{R}^n$. The associated local trajectory is collision-free, satisfies the motion constraints and minimizes the deviation to a preferred velocity $\bar{\mathbf{u}}$.

We approximate robots by their smallest enclosing cylinder of radius r and height $2h$, denoted by V, and its ε-additive dilation of radius $\bar{r} = r + \varepsilon$ and height $\bar{h} = h + \varepsilon$ by V_ε, and assume that all dynamic obstacles maintain a constant velocity during the planning horizon, or cooperate in avoiding collisions.

3 Problem Formulation and Approach

Example 1 Consider the workspace in Fig. 1a, where two robots are tasked with visiting regions G1 and G2 infinitely often, $\varphi_s^g = \bigwedge_{i\in\{1,2\}} \square\diamondsuit(\pi^i_{G1}) \wedge \square\diamondsuit(\pi^i_{G2})$.

Figure 1 shows three approaches for solving this task. A simplistic approach is given in (a), where the robots do not have any collision avoidance and must always be one region apart from one another. The result is thus conservative; in fact, if any one region is blocked, the spec would be unrealizable. As will be shown in Sect. 7 this approach does not scale in the presence of dynamic obstacles. In (b), the robots employ a local planner to avoid collisions, along with a high-level controller that is less conservative but ignores deadlock. In this case, the execution fails to satisfy the task when the two robots become deadlocked. (c) shows our approach, where both robots are able to resolve encountered deadlocks under the synthesized integrated controller. The strategy can exploit the use of other regions if deadlock occurs.

3.1 Problem Formulation

Problem 1 (*Local Collision Avoidance*) For each robot of the team, construct an online local planner that respects the dynamics of the robot and guarantees collision avoidance with static and dynamic (moving) obstacles.

Problem 2 (*Synthesis of High-level Controller with Deadlock Resolution*) Given a topological map, a local motion planner that solves Problem 1 and a realizable mission specification φ that ignores collisions, automatically construct a specification φ' that models deadlock between robots and unmodeled dynamic obstacles and synthesize a controller that satisfies φ'.

By solving Problem 2, we guarantee avoiding deadlocks, but possibly at the sacrifice of task fulfillment. We therefore synthesize *environment assumption revisions*

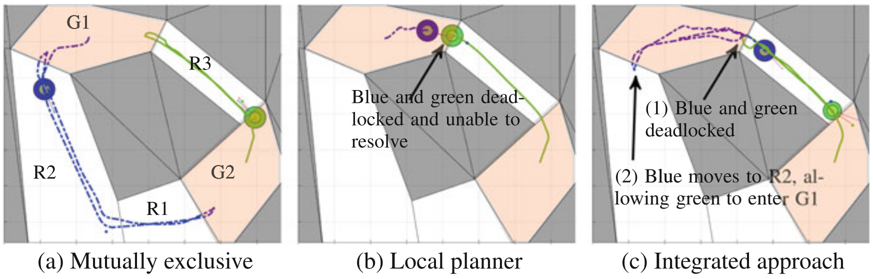

G1
R3
R2
G2
R1
Blue and green deadlocked and unable to resolve
(1) Blue and green deadlocked
(2) Blue moves to R2, allowing green to enter G1
(a) Mutually exclusive
(b) Local planner
(c) Integrated approach

Fig. 1 Three examples of motion planning, where the *blue* and *green* robots are initially in G1 and G2, respectively. **a** is the case with a global planner with no local planner; **b** and **c** correspond to the specifications φ (no deadlock resolution) and φ'' (with deadlock resolution) respectively

(additional LTL formulas) to identify detrimental cases where dynamic obstacles may trigger deadlock and trap the system from achieving its goals. These formulas are significant because they offer conditions upon which the environment must adhere to in order for the robot team to guarantee the task. As such, they must be clearly explained to the user. An example of such a condition is: "the environment will never cause deadlock if robot 1 is in the kitchen and moving to the door".

Problem 3 (***Revising Environment Assumptions***) Given an *unrealizable* reactive mission specification φ', synthesize environment assumption revisions $[\varphi_t^e]^{rev}$ such that the specification φ'' formed by replacing φ_t^e with $[\varphi_t^e]^{rev}$ is realizable, and provide the user with a human-readable description of these revisions.

3.2 Approach

We present a two-part solution to Problems 1, 2 and 3. Figure 2 shows the offline and online components and their interconnections; we now describe them in detail.

Offline. Given a high-level specification that takes a discrete topological map of the workspace and ignores collisions, a centralized controller is synthesized that considers possible deadlocks, iteratively revising the environment assumptions as necessary until a such a controller is synthesized. We also adopt a recovery scheme [17] that synthesizes a strategy that allows violations of environment *safety* assumptions to be tolerated, retaining satisfaction guarantees as long as the violation is transient. The automaton is agnostic to the robot's dynamics, which are instead accounted for by the local planner. The offline high-level synthesis is described in Sect. 4.

Online. At each time step of the execution, the synthesized automaton provides a desired goal for each controlled robot. Each robot independently computes a local trajectory that achieves its goal while avoiding other agents. If a deadlock is sensed, an alternative goal is extracted for some robot; the existence of such an alternative in the automaton is guaranteed by construction. The online local planner builds on [2] by adopting a convex optimization approach as described in Sect. 5.

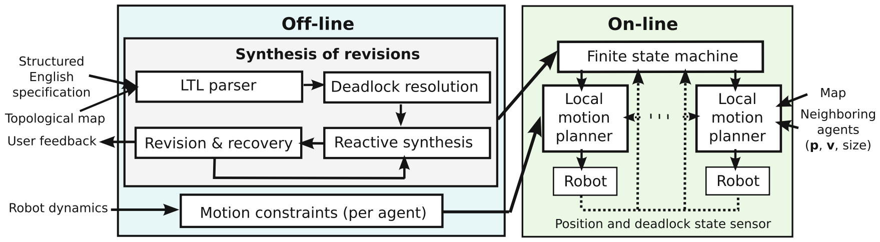


Fig. 2 Structure of the proposed mission planner, with offline and online parts

4 Offline Synthesis of the High-Level Mission Plan

We consider a high-level task specification, and a topological map of the environment composed of regions. This task specification φ ignores collisions and may result in deadlocks as in Fig. 1b. We modify the input specification with additional behaviors that the robot can take to resolve deadlock. The synthesized automaton guarantees completion of the task while redirecting the robots whenever deadlock occurs.

4.1 Deadlock Resolution

We define physical deadlock to be a situation where at least one robot has not reached its goal but cannot move. This can happen when an agent becomes blocked either by another agent or by a dynamic obstacle. To allow the high-level controller to resolve deadlock, we define a Boolean input signal $x^i \in \mathcal{X}$ that declares when a robot is in deadlock, where $i = 1, \ldots, n_{robots}$. We use the term *singleton deadlock* to refer to the specific case where a robot is in proximity of a dynamic obstacle. Additionally, define $x^{ij} \in \mathcal{X}$ to be an input signal that is `True` when two robots are in *pairwise deadlock* (both in deadlock and within a certain distance of one another), and `False` otherwise. We introduce the following shorthand:

$$\begin{aligned}
\theta_P^{ij} &= \neg x^{ij} \wedge \bigcirc x^{ij} && \text{rising edge–pairwise deadlock between robots } i \text{ and } j\\
\theta_S^{i} &= \neg x^{i} \wedge \bigcirc x^{i} && \text{rising edge–singleton deadlock for robot } i\\
\psi_{\alpha\beta}^{i} &= \pi_\alpha^i \wedge \bigcirc \pi_\alpha^i \wedge \pi_{act,\beta}^i && \text{incomplete transition } (\alpha \neq \beta)\text{; remain in region } (\alpha = \beta)
\end{aligned}$$

Resolving deadlock by redirecting the robot's motion based on the instantaneous value of x^i or x^{ij} alone may result in livelock, where the robot may be trapped away from its goals as a result of an alternating deadlock status. For this reason, our scheme automatically introduces additional memory propositions that are set when deadlock is sensed, and reset once the robot leaves its current region. While adding these propositions increases the state space of the synthesis problem, the advantage is that the robot can remember that deadlock had occurred and actively alter its strategy to overcome that situation. For each robot, we introduce the system propositions $\{y_\beta^i \mid R_\beta \in \mathcal{R}\} \subset \mathcal{Y}$ to represent the *memory* of deadlock occurring when activating a transition from a given region to region R_β.

$$\square \bigwedge_{\substack{\pi_\alpha^i \in AP_{\mathcal{R}},\\ R_\beta \in Adj(R_\alpha)}} \left(y_\beta^i \wedge \pi_\alpha^i \implies \bigcirc(\neg\pi_{act,\alpha}^i \wedge \neg\pi_{act,\beta}^i)\right). \tag{2}$$

The role of y_β^i is to disallow the current transition (from R_α to R_β), as well as the self-transition from R_α to R_α. The self-transition is disallowed to force the robot to leave the region where the deadlock occurred (R_α), instead of waiting for it to resolve; R_β is disallowed since the robot cannot make that transition.

Next, we enforce conditions to retain memory of *singleton deadlock*:

$$\bigwedge_{\substack{\pi_\alpha^i \in AP_{\mathcal{R}},\\ R_\beta \in Adj(R_\alpha)}} \left(\neg y_\beta^i \Rightarrow \left((\theta_S^i \wedge \psi_{\alpha\beta}^i) \Rightarrow \bigcirc y_\beta^i\right)\right) \text{ and } \bigwedge_{\substack{\pi_\alpha^i \in AP_{\mathcal{R}},\\ R_\beta \in Adj(R_\alpha)}} \left(y_\beta^i \Rightarrow \left((\pi_\alpha^i \wedge \bigcirc \pi_\alpha^i) \Leftrightarrow \bigcirc y_\beta^i\right)\right). \tag{3}$$

The first formula sets the memory of deadlock y_β^i if the robot is activating transition from R_α to R_β. The second formula keeps memory set until a transition has been made out of R_α (to a region different from R_β).

For *pairwise deadlock*, we add the following conditions to set the memory proposition for at least one robot (at least one of the two robots reacts to the deadlock):

$$\Box\left(\theta_P^{ij} \implies \left(\bigvee_{\ell \in \{i,j\}} \bigwedge_{\substack{\pi_\alpha^\ell \in AP_{\mathcal{R}},\\ R_\beta \in Adj(R_\alpha)}} (\neg y_\beta^\ell \wedge \psi_{\alpha\beta}^\ell) \implies \bigcirc y_\beta^\ell\right)\right). \tag{4}$$

We also add the following to ensure that the memory propositions are only set when the rising edge of deadlock (singleton or pairwise) is sensed.

$$\Box\left(\bigwedge_{\substack{i \in [1, n_{robots}]\\ R_\beta \in \mathcal{R}}} \left(\neg y_\beta^i \wedge \neg\theta_S^i \wedge \bigwedge_{\substack{j \in [1, n_{robots}]\\ j \neq i}} \neg\theta_P^{ij}\right) \implies \bigcirc \neg y_\beta^i\right). \tag{5}$$

In practice, we do not need a proposition y_β^i for every $\mathcal{R}_\beta \in \mathcal{R}$, but only $d = \max_{R_\alpha \in \mathcal{R}} (|Adj(R_\alpha)|)$ such propositions for each robot in order to remember all of the deadlocks around each region of the workspace. The number of conjuncts required for condition (4) is $\binom{n_{robots}}{2}$, but this has no effect on scalability since the runtime of the synthesis algorithm is only a function of the number of propositions and not the size of the specification.

Conjuncting the conditions (2)–(5) with φ_t^s yields a modified formula $[\varphi_t^s]'$ over the set AP, and the new specification $\varphi' = \varphi_i^e \wedge \varphi_t^e \wedge \varphi_g^e \implies [\varphi_i^s]' \wedge [\varphi_t^s]' \wedge \varphi_g^s$, where the initial conditions are modified by setting additional propositions x^i, y_α^i to false.

4.2 *Specification Revisions*

If the above specification φ' is synthesizable, Problem 2 is solved (for a proof, see Sect. 6). However, if the added restrictions to the system behavior result in the specification being unrealizable, Problem 3 must be solved by finding a set of assumptions on deadlock under which the environment must follow for the task to be guaranteed.

These assumptions are then presented to the user as a certificate of the conditions under which the guarantees for deadlock and livelock avoidance hold.

When a specification is unrealizable, there exist environment behaviors (called *environment counterstrategies*) that prevent the system from achieving its goals safely. Here we build upon the work of [3, 6, 11], processing synthesized counterstrategies to mine the necessary assumptions. Rather than synthesize assumptions from the counterstrategy, we instead search the counterstrategy for all deadlock occurrences, then store the corresponding conditions as assumptions.

We denote $\mathcal{C}_{\varphi'}$ as a state machine representing the counterstrategy for φ' and $\mathcal{Q}$ as the set of states for $\mathcal{C}_{\varphi'}$. To find the graph cuts in the counterstrategy graph that prevent the environment from impeding the system, we first define the following propositional representation of state $q \in \mathcal{Q}$ as $\psi(q) = \psi_{\mathcal{X}}(q) \wedge \psi_{\mathcal{Y}}(q)$, where

$$\psi_{\mathcal{Y}}(q) = \bigwedge_{\pi \in \gamma_{\mathcal{Y}}(q)} \pi \wedge \bigwedge_{\pi \in \mathcal{Y} \setminus \gamma_{\mathcal{Y}}(q)} \neg\pi, \quad \psi_{\mathcal{X}}(q) = \bigwedge_{\pi \in \gamma_{\mathcal{X}}(q)} \pi \wedge \bigwedge_{\pi \in \mathcal{X} \setminus \gamma_{\mathcal{X}}(q)} \neg\pi.$$

Next, the set of *cut transitions* S_{cuts} is computed as $S_{cuts} = \{(p, q) \in \mathcal{Q}^2 \mid q \in \delta(p), \psi(p)\psi(q) \models \bigvee_{i \in [1, n_{robots}]} \bigcirc \theta_S^i\}$, where $\delta : \mathcal{Q} \times 2^{\mathcal{Y}} \to 2^{\mathcal{Q}}$ is a transition relation returning the set of possible successor states given the current state and valuations of robot commands in $\mathcal{Y}$. S_{cuts} collects those transitions on which the environment has intervened (by setting deadlock) to prevent the system from reaching its goals.

Finally, the following safety assumptions are found:

$$\varphi_{rev}^e = \square \bigwedge_{(p,q) \in S_{cuts}} (\psi_{\mathcal{Y}}(p) \wedge \psi_{\mathcal{X}}(p) \implies \neg \bigcirc \psi_{\mathcal{X}}(q)) \tag{6}$$

If any of the conjuncts in (6) falsify the environment, they are discarded. Then, set $[\varphi_t^e]^{rev} = \varphi_t^e \wedge \varphi_{rev}^e$ and construct the final specification $\varphi'' = \varphi_i^e \wedge [\varphi_t^e]^{rev} \wedge \varphi_g^e \implies [\varphi_i^s]' \wedge [\varphi_t^s]' \wedge \varphi_g^s$.

Algorithm 1 expresses our proposed approach for resolving deadlock. The automatically generated assumptions act to restrict the behavior of the dynamic obstacles. Each revision of the high-level specification excludes at least one environment move in a given state. Letting $|\cdot|$ denote set cardinality, with $2^{|\mathcal{X}|}$ environment actions and $2^{|\mathcal{Y}|}$ states, at most $2^{(|\mathcal{Y}|+|\mathcal{X}|)}$ iterations occur, though in our experience far fewer are needed. The generated assumptions are minimally restrictive – omitting even one allows the environment to cause deadlock, resulting in unrealizability.

Algorithm 1 Find realizable φ'' fulfilling task φ and resolving deadlock

```
1: $\varphi' \leftarrow \varphi_i^e \wedge \varphi_t^e \wedge \varphi_g^e \implies [\varphi_i^s]' \wedge [\varphi_t^s]' \wedge \varphi_g^s$
2: $[\varphi_t^e]^{rev} \leftarrow \varphi_t^e$;   $\varphi'' \leftarrow \varphi_i^e \wedge [\varphi_t^e]^{rev} \wedge \varphi_g^e \Rightarrow [\varphi_i^s]' \wedge [\varphi_t^s]' \wedge \varphi_g^s$
3: while $\varphi''$ is unrealizable do
4:     Extract $\mathcal{C}_{\varphi''}$ from $\varphi''$
5:     $\varphi_{rev}^e \leftarrow$ Eq. (6)
6:     for each $k$th conjunct of $\varphi_{rev}^e$ s.t. $\varphi_{rev}^e[k] \wedge [\varphi_t^e]^{rev} \neq \texttt{False}$ do
7:         Parse $\varphi_{rev}^e[k]$ into human-readable form and display to user.
8:         $[\varphi_t^e]^{rev} \leftarrow [\varphi_t^e]^{rev} \wedge \varphi_{rev}^e[k]$;   $\varphi'' \leftarrow \varphi_i^e \wedge [\varphi_t^e]^{rev} \wedge \varphi_g^e \Rightarrow [\varphi_i^s]' \wedge [\varphi_t^s]' \wedge \varphi_g^s$
9:     end for
10: end while
```

5 Online Local Motion Planning

The result of the computation of Sect. 4 is a finite state machine where the states are labeled by regions and the transitions represent actions within the allowed navigation path. Each robot executes the finite state machine controller such that the overall multi-robot system is guaranteed to be livelock and collision free and deadlocks are resolved (may they appear). In this section we describe the local planner that links the high-level mission plan with the physical robot (recall Fig. 2). At each step of the online execution, the synthesized finite state machine provides a desired goal position for each robot and a preferred velocity $\bar{\mathbf{u}} \in \mathbb{R}^n$ towards it.

5.1 Constraints

To define the motion and inter-agent avoidance constraints we build on the approach in [2]. We additionally introduce constraints for avoiding static obstacles. For completeness, we give an overview of each of the constraints.

Motion constraints. Recalling Eq. (1) the motion constraint is given by the reference velocities for which the tracking error is below ε, $R(\mathbf{z}, \varepsilon) = \{\mathbf{u} \mid \gamma(\mathbf{z}, \mathbf{u}) \leq \varepsilon\}$, approximated by the largest inscribed convex polytope/ellipsoid $\hat{R}(\mathbf{z}, \varepsilon) \subset R(\mathbf{z}, \varepsilon)$.

Avoidance of other agents. Denote by $\mathbf{p}_j$, $\mathbf{v}_j$, $\bar{r}_j$ and $\bar{h}_j$ the position, velocity, dilated radius and height of a neighboring agent j. Assume that it keeps its velocity constant for $\tilde{t} \leq \tau$, for reciprocity see [2]. For every neighboring agent j, the constraint is given by the reference velocities $\mathbf{u}$ for which the agents' enveloping shape do not intersect within the time horizon. For cylindrically-shaped agents moving in 3D the velocity obstacle of colliding velocities is a truncated cone $VO_j^\tau =$

$$\{\mathbf{u} \mid \exists \tilde{t} \in [0, \tau] : \|\mathbf{p}^H - \mathbf{p}_j^H + (\mathbf{u}^H - \mathbf{v}_j^H)\tilde{t}\| \leq \bar{r} + \bar{r}_j,\ |p^V - p_j^V + (u^V - u_j^V)\tilde{t}| \leq \bar{h} + \bar{h}_j\},$$

where $\mathbf{p} = [\mathbf{p}^H, p^V]$, with $\mathbf{p}^H \in \mathbb{R}^2$ its projection onto the horizontal plane and $p^V \in \mathbb{R}$ its vertical component. The constraint is linearized to $A_j(\mathbf{p}, \varepsilon) = \{\mathbf{u} \mid \mathbf{n}_j^T \mathbf{u} \leq b_j\}$, where $\mathbf{n}_j \in \mathbb{R}^3$ and $b_j \in \mathbb{R}$ maximize $\mathbf{n}_j^T \mathbf{v} - b_j$ subject to $A_j(\mathbf{p}, \varepsilon) \cap VO_j^\tau = \emptyset$.

Avoidance of static obstacles. We extend a recent fast iterative method to compute the largest convex polytope in free space [7], by directing the growth of the region in the preferred direction of motion and enforcing that both the current position of the robot and a look ahead point in the preferred direction of motion are within the region. The convex polytope is computed in position space ($\mathbb{R}^3$ for aerial vehicles) and then converted to an equivalent region in reference velocity space. The details are given in Algorithm 2, where *directedEllipsoid*$(\mathbf{p}, \mathbf{q})$ is the ellipsoid with one axis given by the segment $\mathbf{p} - \mathbf{q}$ and the remaining axis infinitesimally small.

Algorithm 2 Largest collision-free directed convex polytope

```
1: $L \leftarrow \mathbf{p} + \bar{\mathbf{u}}\{\tau, 0.7\tau, 0.5\tau, ..., 0\}$ ;  $\mathbf{q} \leftarrow L[0]$;   $L := L \setminus \mathbf{q}$;   $P := \emptyset$
2: while $L \neq \emptyset$ and $\mathbf{p}, \mathbf{q} \notin P$ do
3:     $E \leftarrow directedEllipsoid(\mathbf{p}, \mathbf{q})$
4:     while not converged do // Largest polytope seeded in $E$ computed as in [7]
5:         $P \leftarrow$ separating planes of $E$ and dilated $\mathcal{O}$ (Quadratic program) , $P \subset \mathbb{R}^n \setminus (\mathcal{O} + V_\varepsilon)$
6:         If $\mathbf{p}, \mathbf{q} \notin P$ then { $\mathbf{q} \leftarrow L[0]$;  $L := L \setminus \mathbf{q}$;  break; }
7:         $E \leftarrow$ ellipsoid $E \subset P$ of maximal volume (Semi-Definite Program)
8:     end while
9: end while
10: $F(\mathbf{p}, \varepsilon) := (P - \mathbf{p})/\tau$ // Converts to ref. velocity, $\mathbf{u}$, space
```

5.2 *Optimization*

The optimization cost is given by two parts. The first part is a regularizing term penalizing changes in velocity (weighted by design constant $\bar{\alpha}$); the second minimizes the deviation to a preferred velocity, corrected by a repulsive velocity $\mathring{\mathbf{u}}$ inversely proportional to the distance to neighboring obstacles [2] when in close proximity. A convex optimization with quadratic cost and mixed linear/quadratic constraints is solved:

$$\mathbf{u}^* := \underset{\mathbf{u} \in \mathbb{R}^n}{\arg\min}\ (\bar{\alpha}||\mathbf{u} - \mathbf{v}||^2 + ||\mathbf{u} - (\bar{\mathbf{u}} + \mathring{\mathbf{u}})||^2), \tag{7}$$
$$\text{s.t.} \quad \mathbf{u} \in \hat{R}(\mathbf{z}, \varepsilon) \cap F(\mathbf{p}, \varepsilon) \quad \text{and} \quad \mathbf{n}_j^T \mathbf{u} \leq b_j \quad \forall j \text{ neighboring agent}$$

The solution of this optimization is a collision-free reference velocity $\mathbf{u}^*$ which minimizes the deviation towards the goal specified by the high-level state machine. The associated trajectory (see Sect. 2.3) is followed by the robot and is collision-free.

6 Guarantees

We provide proofs for the guarantees inherent to our synthesized controller.

Respects the modeled robot dynamics. By construction of the local planner, the controller is guaranteed correct with respect to the low-level controller $f(\mathbf{z}, \mathbf{u}, \tilde{t})$, which is continuous on the initial state of the robot and respects its dynamics.

Yields collision-free motion. If (7) is *feasible*, collision-free motion is guaranteed for the local trajectory up to time τ (the optimal reference trajectory is collision-free for an agent whose volume is enlarged by ε and the robot stays within ε of it) with the assumption that all interacting agents maintain a constant velocity. Avoidance of dynamic obstacles was shown by [2], we reproduce it for the case of a dynamic obstacle in $\mathbb{R}^2$ (through it is extensible to $\mathbb{R}^3$). Let $\mathbf{p}(t)$ denote the position at time $t \geq t^k$. If not specified, variables are evaluated at t^k. Consider $||\mathbf{p}(t) - \mathbf{p}_j(t)|| =$

$$||f(\mathbf{z}, \mathbf{u}, \tilde{t}) - (\mathbf{p}_j + \mathbf{v}_j\tilde{t})|| \underset{\mathbf{u}\in\hat{R}(\mathbf{z},\varepsilon)}{\geq} ||(\mathbf{p} + \mathbf{u}\tilde{t}) - (\mathbf{p}_j + \mathbf{v}_j\tilde{t})|| - \varepsilon \underset{\mathbf{u}\in A_j(\mathbf{p},\varepsilon)}{\geq} r + r_j$$

For avoidance of static obstacles, we have that $\mathbf{u} \in F(\mathbf{p}, \varepsilon)$ implies, for all $\tilde{t} \in [0, \tau]$,

$$\mathbf{u} \in F(\mathbf{p}, \varepsilon) \underset{\text{Alg. 2, } P \text{ convex}}{\Rightarrow} (\mathbf{p} + \mathbf{u}\tilde{t}) \notin \mathcal{O} + V_\varepsilon \underset{\mathbf{u}\in\hat{R}(\mathbf{z},\varepsilon)}{\Rightarrow} f(\mathbf{z}, \mathbf{u}, \tilde{t}) \notin \mathcal{O} + V.$$

If (7) is *infeasible*, no collision-free solution exists that respects all of the constraints. Since the time horizon is larger than the required time to stop, passive safety is preserved by slowing down on the last feasible path and eventually reaching a stop. Also, since this computation is performed at a high frequency, each individual robot is able to adapt to changing situations, and the resulting motion is collision-free.

Realizes the reactive task specification. Since the local planner is myopic, it provides guarantees up to a time horizon τ and consequently may result in deadlock and livelock. However, as we have shown, the planner's local guarantees allow a discrete abstraction that the high-level strategy can use to resolve deadlocks and avoid livelocks. Here we formally prove the guarantees on the high-level behavior provided by our synergistic online and offline synthesis.

Proposition 1 *Given a task specification φ that ignores collisions, if the resulting specification φ' defined in Sect. 4 is realizable, then the corresponding strategy automaton also realizes φ.*

Proof Assume given $\varphi = \varphi_i^e \wedge \varphi_t^e \wedge \varphi_g^e \implies \varphi_i^s \wedge \varphi_t^s \wedge \varphi_g^s$. Recall that $\varphi' = \varphi_i^e \wedge \varphi_t^e \wedge \varphi_g^e \implies [\varphi_i^s]' \wedge [\varphi_t^s]' \wedge \varphi_g^s$, where $[\varphi_i^s]'$ and $[\varphi_t^s]'$ contain φ_i^s and φ_t^s as subformulas, respectively. Suppose that strategy automaton $\mathcal{A}_{\varphi'}$ realizes φ'. This means that the resulting controller is guaranteed to fulfill the requirement $[\varphi_i^s]' \wedge [\varphi_t^s]' \wedge \varphi_g^s$ as long as the environment fulfills the assumption $\varphi_i^e \wedge \varphi_t^e \wedge \varphi_g^e$. This implies that $A_{\varphi'}$ fulfills $\varphi_i^s \wedge \varphi_t^s \wedge \varphi_g^s$ as long as the environment fulfills the assumption $\varphi_i^e \wedge \varphi_t^e \wedge \varphi_g^e$. □

Proposition 2 *Given a task specification φ that ignores collisions, if φ is realizable but the resulting specification φ' is not realizable, then the revision procedure in Sect. 4.2 will find an assumption φ_{rev}^e to add to φ'.*

Proof Suppose φ is realizable by strategy $\mathcal{A}_\varphi$, but φ' is not realizable, admitting counterstrategy $\mathcal{C}_{\varphi'} = (\mathcal{Q}, \ldots)$. It suffices to show that the set S_{cuts} is nonempty. Assume for a contradiction that S_{cuts} is empty. Then the rising edge of deadlock θ_s^i

never occurs for any i, so no robot transitions are ever disabled. Since we assume that deadlock does not occur in the initial state, this means that x^i is always `False` for every i. Therefore $[\varphi_i^s]' \wedge [\varphi_t^s]' \wedge \varphi_g^s$ defined in Sect. 4 reduces to $\varphi_i^s \wedge \varphi_t^s \wedge \varphi_g^s$. The lack of deadlock means that any region transition contained in $\mathcal{A}_\varphi$ is still admissible, and therefore $\mathcal{A}_\varphi$ can be used as a strategy to realize φ'.

Note that it may be the case that S_{cut} is nonempty, but for every $(p, q) \in S_{cuts}$, the resulting revision $(\psi_{\mathcal{Y}}(p) \wedge \psi_{\mathcal{X}}(p) \implies \neg \bigcirc \psi_{\mathcal{X}}(q))$ contradicts φ_e^t. This indicates that φ is only realizable because it makes unreasonable assumptions on the environment. Our approach identifies this fact as a by-product of the revision process.

Computational complexity. The high-level reactive synthesis is exponential in the number of propositions [4], which scales linearly with n_{robots} – no worse than existing approaches (e.g. [15]). When one or more dynamic obstacles are considered, the number of propositions does not depend on the number of dynamic obstacles.

For the online component, a convex program is solved independently for each robot, with the number of constraints linear in the number of neighboring robots. The run-time of the iterative computation of the convex volume in free space barely changes with the number of obstacles, up to tens of thousands [7], and a timeout can be set, with the algorithm returning the best solution found.

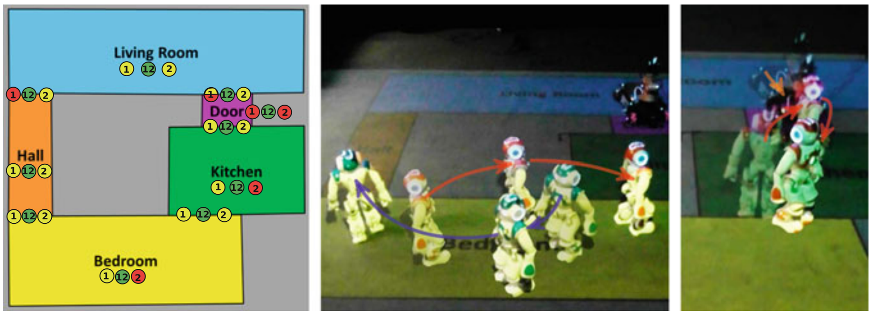
7 Experiments and Simulations

The synthesis procedure described in Sect. 4 was implemented with the `slugs` synthesis tool [8], and executed with the LTLMoP toolkit [9]. The local motion planner, Sect. 5, was implemented with the IRIS toolbox [7] and an off-the-shelf convex optimizer. We consider the dynamic obstacles to be cooperative in avoiding collisions. A video is available at http://youtu.be/esa3osYtvGA.

7.1 Humanoid Robots

We synthesize a controller for a "garbage collection" scenario, carried out by two humanoid robots (able to rotate in place, move forward and along a curve) occupying the workspace in Fig. 3a. The robots are required to patrol the *Living Room* (R_{LR}) and *Bedroom* (R_{BR}) [$\square\diamond(\pi_{LR}^1) \wedge \square\diamond(\pi_{BR}^1) \wedge \square\diamond(\pi_{LR}^2) \wedge \square\diamond(\pi_{BR}^2)$] and if *garbage* is observed, pick it up [$\square(\pi_{garb}^1 \implies \pi_{act,pickup}^1) \wedge \square(\pi_{garb}^2 \implies \pi_{act,pickup}^2)$]. The robots must always avoid other moving agents.

The system propositions are actions to move between regions $(\pi_{act,LR}^i, \ldots, \pi_{act,BR}^i)$ and to pick up $(\pi_{act,pickup}^i)$. The environment propositions are sensed garbage (π_{garb}^i), region completions $(\pi_{LR}^i, \ldots, \pi_{BR}^i)$, and pick up completion (π_{pickup}^i). We omit the encoding of Definition 2, though these conditions are implied.



(a) Workspace with specification revisions

(b) Avoidance maneuver

(c) Deadlock resolution

Fig. 3 Planar scenario with two centrally-controlled Nao robots and a dynamic obstacle (youBot). **a** Workspace showing specification revisions for each region completion/activation pairs where singleton or pairwise deadlock may occur. Dot placement corresponds to the $AP_{\mathcal{R}}^{act}$ for each region; numbers indicate the robot(s) that are allowed to be in deadlock, and color represents the necessary restrictions on deadlock. *Green* dots represent transitions where deadlock is allowed; *yellow dots* where deadlock is allowed, but only up to a finite time; and *red dots* where deadlock is not allowed. **b** and **c** Three consecutive frames of the video are superimposed. In (**c**), one of the Naos reverses direction to resolve the deadlock with the youBot

Our synthesis tool took 84 seconds yielding an automaton with 5836 states and four memory propositions.

A graphical representation of the revisions produced by our algorithm is shown in Fig. 3a. The red dots indicate that dynamic obstacles should not produce deadlock when the robot is making the indicated transition. We also alert the user via textual feedback – one of the generated statements for our scenario is: `Deadlock should not occur when robot 1 is in the Hall moving toward the Living Room`. We employ two Aldebaran Nao robots and a teleoperated KUKA youBot as the dynamic obstacle. As demonstrated in the snapshots in Fig. 3, the Naos are capable of executing the task, avoiding collision and resolving deadlocks.

We further evaluated the approach in simulation, from 10 different initial conditions and with two dynamic obstacles. No collisions occurred and the approach was able to resolve 47 deadlock events out of the 51 encountered. Those that could not be resolved occurred in disallowed transitions labeled red in Fig. 3a.

7.2 *Scalability with Respect to Dynamic Obstacles*

Considering the example in Sect. 7.1, the synthesized controller for two robots consists of 29 propositions, and is invariant to the number of dynamic obstacles. In comparison, we consider a baseline approach similar to Fig. 1a without a local planner where one-cell separation with other robots and dynamic obstacles (DO) is kept. In this case, 20 propositions are required for zero DO, 25 for one DO, 30 for two DO,

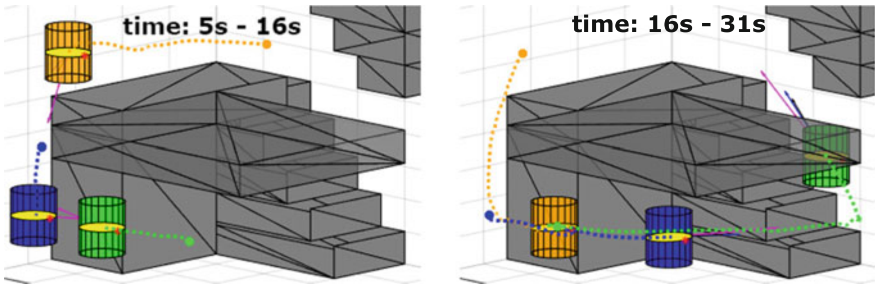


Fig. 4 Deadlock resolution (*green* robot) and safe navigation in a 3D environment. Quadrotors are displayed at the final time and their paths for the time interval. Each *yellow* disk represents a quadrotor and the cylinder its safety volume. The *orange* robot represents the dynamic obstacle

35 for three DO and 80 propositions for eight DO. Because the obstacles are assumed to behave adversarially, they can violate mutual exclusion if they enter to within a neighboring region to the robot. Hence, the synthesis procedure is *not realizable* for one or more dynamic obstacles. Our approach, on the other hand, is realizable independently of the number of dynamic obstacles and requires fewer propositions than the case with two or more DO.

7.3 Quadrotors

We next demonstrate the effectiveness of the approach in a 3D scenario with the $5 \times 5 \times 5\,\mathrm{m}^3$ two floor workspace shown in Fig. 4, where robots can move between floors through a vertical opening at the left corner or the stairs at the right side of the room. We simulate, using the model described in [2], two controlled quadrotors, and one more as a dynamic obstacle. The task is to infinitely often visit the top and bottom floors while avoiding collisions and resolving deadlock. The high-level controller is synthesized as described in Sect. 4. A local planner for the 3D environment is constructed following Sect. 5. A representative experiment is shown in the snapshots in Fig. 4. The green robot enters deadlock when moving towards the upwards corridor; however, deadlock is resolved by taking the alternative route up the stairs.

8 Conclusion

We present a framework for synthesizing a high-level finite state machine and collision-free local planner that guarantees completion of a task specified in linear temporal logic, where we consider high-level specifications that are able to capture basic locomotion, sensing and actuation capabilities. Our approach is less conservative than current approaches that impose a separation between agents, and is computationally cheaper than explicitly modeling all possible obstacles in the environment. If no controller is found that satisfies the specification, the approach automatically generates the needed assumptions on deadlock to render the specification realizable and communicates these to the user. The approach generates controllers that accommodate deadlock between robots or with dynamic obstacles *independently of* the precise number of obstacles present, and we have shown that the generated controllers are correct with respect to the original specification. Experiments with ground and aerial robots demonstrate collision avoidance with other agents and obstacles, satisfaction of a task, deadlock resolution and livelock-free motion. Future work includes optimizing the set of revisions found, and decentralizing the synthesized controller.

Acknowledgements This work was supported in part by NSF Expeditions in Computer Augmented Program Engineering (ExCAPE), ONR MURI Antidote N00014-09-1-1031, SMARTS N00014-09-1051, the Boeing Company and TerraSwarm, one of six centers of STARnet, a Semiconductor Research Corporation program sponsored by MARCO and DARPA.

References

1. Alonso-Mora, J., Gohl, P., Watson, S., Siegwart, R., Beardsley, P.: Shared control of autonomous vehicles based on velocity space optimization. In: 2014 IEEE International Conference on Robotics and Automation (ICRA), pp. 1639–1645 (2014)
2. Alonso-Mora, J., Naegeli, T., Siegwart, R., Beardsley, P.: Collision avoidance for multiple aerial vehicles. Auton. Robot. (2015)
3. Alur, R., Moarref, S., Topcu, U.: Counter-strategy guided refinement of gr(1) temporal logic specifications. In: Formal Methods in Computer-Aided Design (FMCAD), pp. 26–33 (2013)
4. Bloem, R., Jobstmann, B., Piterman, N., Pnueli, A., Sa'ar, Y.: Synthesis of reactive (1) designs. J. Comput. Syst. Sci. **78**(3), 911–938 (2012)
5. Chen, Y., Ding, X.C., Stefanescu, A., Belta, C.: Formal approach to the deployment of distributed robotic teams. IEEE Trans. Robot. **28**(1), 158–171 (2012)
6. DeCastro, J.A., Ehlers, R., Rungger, M., Balkan, A., Tabuada, P., Kress-Gazit, H.: Dynamics-based reactive synthesis and automated revisions for high-level robot control. In: CoRR (2014)
7. Deits, R., Tedrake, R.: Computing large convex regions of obstacle-free space through semidefinite programming. In: Workshop on the Algorithmic Fundamentals of Robotics (2014)
8. Ehlers, R., Finucane, C., Raman, V.: Slugs gr(1) Synthesizer (2013). http://github.com/ltlmop/slugs
9. Finucane, C., Jing, G., Kress-Gazit, H.: Ltlmop: Experimenting with language, temporal logic and robot control. In: Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (2010)
10. Kress-Gazit, H., Fainekos, G.E., Pappas, G.J.: Temporal logic based reactive mission and motion planning. IEEE Trans. Robot. **25**(6), 1370–1381 (2009)
11. Li, W., Dworkin, L., Seshia, S.A.: Mining assumptions for synthesis. In: 9th IEEE/ACM International Conference on Formal Methods and Models for Codesign, MEMOCODE (2011)
12. Livingston, S.C., Prabhakar, P., Jose, A.B., Murray, R.M.: Patching task-level robot controllers based on a local μ-calculus formula. In: Proceedings of the IEEE International Conference on Robotics and Automation (ICRA). Karlsruhe, Germany (2013)
13. Raman, V., Kress-Gazit, H.: Explaining impossible high-level robot behaviors. IEEE Trans. Robot. **29**(1), 94–104 (2013). doi:10.1109/TRO.2012.2214558

14. Raman, V., Piterman, N., Kress-Gazit, H.: Provably correct continuous control for high-level robot behaviors with actions of arbitrary execution durations. In: IEEE International Conference on Robotics and Automation (2013)
15. Ulusoy, A., Smith, S.L., Ding, X.C., Belta, C., Rus, D.: Optimality and robustness in multi-robot path planning with temporal logic constraints. I. J. Robot. Res. **32**(8), 889–911 (2013)
16. Vardi, M.Y.: An automata-theoretic approach to linear temporal logic. In: Logics for concurrency, pp. 238–266. Springer, Heidelberg (1996)
17. Wong, K.W., Ehlers, R., Kress-Gazit, H.: Correct high-level robot behavior in environments with unexpected events. In: Proceedings of Robotics: Science and Systems (2014)
18. Wongpiromsarn, T., Ulusoy, A., Belta, C., Frazzoli, E., Rus, D.: Incremental synthesis of control policies for heterogeneous multi-agent systems with linear temporal logic specifications. In: IEEE International Conference on Robotics and Automation (ICRA) (2013)

An Optimal Control Approach to Mapping GPS-Denied Environments Using a Stochastic Robotic Swarm

Ragesh K. Ramachandran, Karthik Elamvazhuthi and Spring Berman

1 Introduction

In recent years, there has been an increasing focus on the development of robot platforms that can be deployed in *swarms* to perform tasks autonomously over large spatial and temporal scales. In addition, swarms of nanoscale structures and devices such as nanoparticles, molecular machines, and magnetic nanocarriers are being developed for biomedical applications such as imaging and targeted drug delivery [21]. Many potential applications for robotic swarms, including exploration, environmental monitoring, disaster response, search-and-rescue, mining, and intelligence-surveillance-reconnaissance, will require the robots to operate in dynamic, uncertain environments. Moreover, the robots' highly restricted onboard power may preclude the use of GPS and communication devices, or the robots may be located in GPS-denied environments where communication is impractical or unreliable. Despite these limitations, it may still be necessary for the swarm to characterize its surroundings, for instance to map obstacles, target payloads, or hazardous areas to avoid. Nanoscale swarms, which will have extremely limited capabilities, may be used to map cellular structures inside the human body.

To address these challenges, we present a method for *mapping a feature of interest in an unknown environment* using a swarm of robots with local sensing capabilities, no localization, and no inter-robot communication. We consider scenarios where the robots exhibit significant randomness in their motion due to sensor and actuator noise

R.K. Ramachandran (✉) · K. Elamvazhuthi · S. Berman
School for Engineering of Matter, Transport and Energy, Arizona State University, Tempe, AZ, USA
e-mail: rageshkr@asu.edu

K. Elamvazhuthi
e-mail: karthikevaz@asu.edu

S. Berman
e-mail: Spring.Berman@asu.edu

A. Bicchi and W. Burgard (eds.), *Robotics Research*, Springer Proceedings in Advanced Robotics 2, DOI 10.1007/978-3-319-51532-8_29

or, at the nanoscale, the effects of Brownian motion and chemical interactions. Our mapping approach is scalable with the number of robots, so that arbitrary swarm populations can be used.

Our method relies on developing a continuous abstraction of the swarm population dynamics in the form of an advection-diffusion-reaction PDE model, which we call the *macroscopic model*. This model describes the spatial and temporal evolution of the population densities of robots in different states throughout the domain. To represent individual robots, we define a *microscopic model* that describes how each robot moves and responds upon encountering a feature of interest. The state transition of a robot is modeled as a irreversible chemical reaction with a high reaction rate. The macroscopic model becomes a more accurate model of the microscopic model as the number of robots increases.

We pose our mapping problem as the computation of a spatially varying function that represents the map of the feature of interest. To estimate this function, we use temporal data that is recorded by the robots during their exploration of the environment. This data yields the time evolution of the number of robots that are still exploring the domain; i.e., robots that have not encountered the feature. In practice, this data could be collected from the robots after their deployment by retrieving their recorded times of encounter with the feature. In biomedical imaging applications with nanoscale swarms, this data could be obtained from a measurable signal that corresponds to the density of the population that is still in the exploring state.

Once this data is obtained, we use techniques from optimal control to compute the function that represents the feature map. In general, optimal control entails the minimization or maximization of an objective functional that is defined in a finite-dimensional space and is subject to a set of ordinary or partial differential *constraint equations*, which govern the system of interest. From a computational perspective, optimal control methods are more effective than black box techniques, such as genetic algorithms and particle swarm optimization, in terms of the number of objective functional evaluations per cycle. This computational advantage mainly arises from their use of the problem structure to calculate the gradient of the control-to-state maps using the adjoint equation. The feature map is defined as the solution of an optimization problem that minimizes an objective functional which is based on the robot data. This optimization problem is solved numerically offline using standard techniques such as gradient descent algorithms. We validate our approach in simulation for features of varying shape, size, orientation, and location.

1.1 Related Work

In the literature, there have been exhaustive studies on mapping and exploring an environment using robots. SLAM (simultaneous localization and mapping) [16, 18], probabilistic mapping [3, 19], and topological and metric map building [15, 20] are some of the techniques that have been developed for environmental mapping by robots. These techniques have been used for path planning and mapping in small multi-robot groups. However, the problem of scaling these approaches to larger groups becomes intractable for swarms of hundreds or thousands of robots, due to their limitations on communication bandwidth and their spatially distributed nature. In addition, these techniques require the robots to have sophisticated sensing and processing capabilities, which are not feasible in swarm robotic platforms.

Mapping an environment using a robotic swarm is a relatively new area of research in the robotics community. An approach to this problem is given in [6, 7], in which a robotic swarm is used to identify the topological features of an environment from information about the times at which robots encounter other robots and environmental features. This work borrows tools from algebraic geometry and topological data analysis to compute a metric that can be used to classify the topological structure of the environment. The approach requires some minimal inter-robot communication, unlike our strategy which is communication-free.

Our mapping approach uses methods from [9], a stochastic task allocation approach that achieves target spatial distributions of robot activity without using communication or localization. Also, our approach is inspired by [13], a method for reconstructing environmental features from minimal robot data using compressed sensing techniques. In contrast to the scenarios that we consider, the robots in [9, 13] can move over the features to be mapped, which allows the mapping problem to be formulated as the inversion of a linear operator. Approaches with a similar mathematical framework for parameter estimation have been used extensively in the area of biomedical imaging, especially with MRI and CT scan images. In these approaches, the system is excited with a stimulus such as a magnetic field, X-rays, or ultrasound, and the system response is used to identify and estimate a spatially-dependent parameter that corresponds to the image [1, 17, 23].

2 Problem Statement

We consider a scenario in which N robots are deployed into an unknown, bounded environment to map a single feature of interest. We exclude cases in which the feature is located very close to the domain boundary, since robot collisions with this boundary and the high diffusion of swarms that start far from the feature will degrade the estimation. If a robot encounters the feature, it stops moving and records the time at which it stopped. Using data on the number of robots that are still moving at each instant, we aim to estimate the position and geometry of the encountered feature. We can improve the accuracy of this estimate by deploying the swarm in different directions from various locations, which will ensure greater coverage of the domain and result in robot collisions with a larger portion of the feature boundary. This approach may be used to map multiple sparsely distributed features by reconstructing each individual feature from its corresponding data set and computing the entire map as a linear combination of single-feature maps.

Robot capabilities: The robots are assumed have sufficient power to complete the mapping operation. The power requirement for the robots is low, since they are not equipped with communication devices or GPS. The robots have local sensing capabilities and can identify the feature at distances within their sensing range. We may also assume that the robots can detect other robots within their sensing range and perform collision avoidance maneuvers, although we do not simulate collision avoidance in this work. Each robot is equipped with a compass and thus can move in a specified heading. Additionally, the robots have sufficient memory to store the time of their encounter with the feature.

Robot controller: The robots begin at a specified location in the domain. During a swarm deployment, the robots move with a predetermined time-dependent velocity, $\mathbf{v}(t) \in \mathbb{R}^2$. This velocity is designed to guide the center of mass of the swarm along a desired trajectory through the environment. The velocity field may be initially transmitted to the robots by a computer at their starting location, or the robots may be directed according to the field using external stimuli such as magnetic fields or radiation. The robots' motion is affected appreciably by sensor and actuator noise, due to lack of feedback. If a robot detects a feature within its sensing range, it stops moving and records the time. At a predefined time t_f, the stationary robots around the feature boundary return to the starting point of the deployment and upload their encounter times to a computer. The computer then applies the optimal control method described in Sect. 4 to estimate the map of the feature using this robot data.

3 Models of the Mapping Scenario

3.1 Microscopic Model

This model is used to simulate a robot's motion and its response to an encounter with a feature in its path. The change in a robot's state that is triggered by an encounter is modeled as an irreversible chemical reaction,

$$A \xrightarrow{k} P, \tag{1}$$

where the species A represents an *active* (moving) robot, P represents a *passive* (stationary) robot, and k is the reaction rate constant, which in this case is a fixed probability per unit time. This constant is assigned a high value to enforce a high probability of transitioning from active to passive.

We model the robots as point masses with negligible size compared to the area of the domain. A particular robot i has position $\mathbf{X}_i(t) = [x_i(t)\ y_i(t)]^T$ at time t. The deterministic motion of the robot is directed by the time-dependent velocity field $\mathbf{v}(t) = [v_x(t)\ v_y(t)]^T$. The noise in the robot movement is modeled as a Brownian motion that drives diffusion with an associated diffusion coefficient D. We assume that the robots' navigation error can be modeled as diffusive noise and that the value of D can be estimated. The displacement of robot i over a time step Δt is given by the standard-form Langevin equation [11]:

$$\mathbf{X}_i(t + \Delta t) = \mathbf{X}_i(t) + (\sqrt{2D\Delta t})\mathbf{Z}(t) + \mathbf{v}(t)\Delta t, \tag{2}$$

where $\mathbf{Z}(t) \in \mathbb{R}^2$ is a vector of independent standard normal random variables that are generated at time t. The robots avoid collisions with the domain boundary by performing a specular reflection when they encounter this boundary.

3.2 Macroscopic Model

The macroscopic model governs the time evolution of the expected spatial distribution of the robotic swarm. For a swarm whose members move according to Eq. (2), the macroscopic model is given by an advection-diffusion PDE, as described in [5]. Since our microscopic model includes robot state changes that can be represented as chemical reactions, our macroscopic model takes the form of an advection-diffusion-reaction (ADR) PDE. The model is defined over a domain $\Omega \subset \mathbb{R}^2$ with Lipschitz continuous boundary $\partial\Omega$ and over a time interval T. We define $L = \Omega \times [0, T]$ and $\Gamma = \partial\Omega \times [0, T]$. The state of the macroscopic model is the population density field $u(\mathbf{x}, t)$ of active robots in the domain at points $\mathbf{x} \in \Omega$ and times $t \in T$. We specify a spatially varying *indicator function*, $K(\mathbf{x}) : \Omega \to \{0, 1\}$, that equals 0 at points $\mathbf{x}$ where the feature of interest is absent and equals 1 at points where it is present. The reaction term of the macroscopic model is determined by the rate constant k in Eq. (1), which is switched on or off by the indicator function $K(\mathbf{x})$ depending on whether the feature of interest occupies point $\mathbf{x}$. This term models the switching of individual robots from the active state to the passive state when they are in the vicinity of the feature. The advection term of the macroscopic model is governed by the velocity field $\mathbf{v}(t)$ that is defined in the microscopic model.

From the above definition, the macroscopic model is given by:

$$\frac{\partial u}{\partial t} = \nabla \cdot (D\nabla u - \mathbf{v}(t)u) - kK(\mathbf{x})u \quad in \;\; L \tag{3}$$

with the no-flux boundary condition

$$\mathbf{n} \cdot (D\nabla u - \mathbf{v}(t)u) = 0 \quad on \;\; \Gamma, \tag{4}$$

where $\mathbf{n} \in \mathbb{R}^2$ is the outward normal of the boundary $\partial\Omega$. We specify that all robots start in the active state and set the initial condition,

$$u(\mathbf{x}, 0) = u_0, \tag{5}$$

to a Gaussian density centered at a point $\mathbf{x}_0$, which we assume is far from the feature. The macroscopic model is numerically solved using the explicit finite-volume method that is described in [9].

Our approach relies on the close correspondence of the macroscopic model solution to the average swarm density over an ensemble of microscopic model simulations. Therefore, the approach is robust to robot malfunctions and external disturbances as long as these factors do not significantly affect the model correspondence. This implies that the number of failed robots should be small compared to the total swarm size, and that the robots' trajectory drift due to wind, currents, and other environmental influences should be small relative to their modeled motion. In scenarios that violate these conditions, it would be necessary to improve the accuracy of the macroscopic model by estimating the components of $\mathbf{v}$, D, and k that are affected by unmodeled dynamics and disturbances. This is a topic of future work.

4 Optimal Control Approach to Mapping Features

The feature reconstruction problem is framed as an optimal control problem. A gradient descent algorithm is used to compute the optimal control for the problem. An adjoint state equation approach is used to compute the gradient required for the algorithm [4]. The key advantage of this approach is that it derives an explicit formula for the gradient of the objective functional with respect to the control, subject to the constraints. The Hamiltonian and Pontryagin maximum principle can be to used to derive the adjoint equation for finite-dimensional systems. However, in the case of infinite-dimensional systems, the existence of the Hamiltonian has been proven only for a limited class of systems [10]. This motivated us to derive the directional directive of the control-to-state mapping and use the generalized chain rule of differentiation of composite mappings in Banach spaces, as is found in the literature [2, 22]. In order to make the derivatives of certain maps well-defined, an appropriate choice of spaces is made for the parameters and the solutions satisfying the system of differential equations. We present a Lagrangian-based analysis of these derivatives in the Appendices. The proof for the existence of optimal control for the problem is the same as the one shown in [8].

The optimization procedure uses data on the ratio of the number of active robots at each instant of time to the initial number of active robots at the start of the swarm deployment. To ensure sufficient coverage of the domain, the swarm can be deployed from multiple starting positions and directed along different trajectories. Once this data is obtained, the optimization procedure is performed to find the feature map that would produce data that is similar to the data obtained from the deployments. The computational cost increases greatly with the number of data sets (one from each deployment) that are used for optimization, since the number of PDEs to be solved per iteration varies linearly with the data sets. However, we can obtain a better estimate of the feature map with more data. Hence, there is a tradeoff between the computational cost of the optimization and the accuracy of the estimate. In order to resolve this issue, we discard data sets from deployments in which few robots undergo a state transition compared to the other deployments. A paucity of state transitions indicates that the swarm trajectories infrequently intersect the feature. In addition, our procedure can be easily parallelized since the most computationally intensive part is the solution of the PDEs.

The optimal control problem is formulated as follows. Each of the i swarm deployments yields a sequence of times at which active robots encounter the feature and switch to the passive state. From this data, we can determine the fraction $g_i(t) \in L^2([0, T])$ of active robots in the swarm at each time t during deployment i. The solution $u_i(\mathbf{x}, t)$ of the corresponding macroscopic model Eqs. (3)–(5) can be used to compute the integral $\int_\Omega u_i(\mathbf{x}, t)d\mathbf{x}$, the expected fraction of active robots in the domain at time t. We assume that the swarm size is sufficiently large for $g_i(t)$ to closely match this integral if the feature map, represented by the function $K(\mathbf{x})$ in Eq. (3), is known. Therefore, we can frame our optimization objective as the computation of the input $K(\mathbf{x})$ that minimizes the function

$$J_i(u_i) = \frac{1}{2} \left\| \int_\Omega u_i(\mathbf{x}, t)d\mathbf{x} - g_i(t) \right\|^2_{L^2([0,T])}. \tag{6}$$

Suppose that the data from N deployments are selected to compute the optimal controls. The swarm velocity and initial distribution for deployment i are given by $\mathbf{v}_i(t)$ and u_0^i, respectively. The macroscopic model with these parameters is considered to be the i^{th} set of constraints, which we denote by $\Psi_i(u_i, K)$ as in [22]. The solution to this model is given by u_i, and the set of solutions for all N deployments is $\mathbf{u} := \{u_1, u_2, \dots, u_i, \dots, u_N\}$. We define the space of macroscopic model solutions as $U = C([0, T]; L^2(\Omega))$ and the space of admissible input functions as $\Theta_{ad} = \{K(\mathbf{x}) \in L^2(\Omega);\ K_{min} \le K(\mathbf{x}) \le K_{max}\}$. Furthermore, W_i is a weight that quantifies the significance of the data from deployment i relative to data from the other deployments, and λ is the Tikhonov regularization parameter [14]. Using these definitions, we can frame the optimal control problem as:

$$\min_{(\mathbf{u}, K(\mathbf{x})) \in U^N \times \Theta_{ad}} \mathbf{J}(\mathbf{u}, K) = \sum_{i=1}^{N} W_i J_i(u_i) + \frac{\lambda}{2} \|K(\mathbf{x})\|^2_{L^2(\Omega)}, \tag{7}$$

subject to the constraints $\Psi_i(u_i, K)$, $i = 1, \dots, N$.

We must compute the gradient of the objective functional $\mathbf{J}(\mathbf{u}, K)$ with respect to the control inputs in order to perform the gradient descent algorithm for minimizing this functional. We introduce the Lagrangian functional $\mathscr{L}$ and Lagrangian multipliers p_i, with $\mathbf{p} := \{p_1, p_2, \dots, p_i, \dots, p_N\}$:

$$\mathscr{L}(\mathbf{u}, \mathbf{p}, K) = \mathbf{J}(\mathbf{u}, K) + \sum_{i=1}^{N} \langle p_i, \Psi_i(u_i, K) \rangle. \tag{8}$$

The functions p_i, also known as the adjoint variables, express the sensitivity of the objective functional to variations in the input control variable $K(\mathbf{x})$. The necessary condition for optimality is $\nabla\mathscr{L} = 0$, which implies the following three conditions: (1) $\nabla_{\mathbf{u}}\mathscr{L} = 0$, the adjoint equation; (2) $\nabla_{\mathbf{p}}\mathscr{L} = 0$, the state equation in weak form; and (3) $\nabla_K\mathscr{L} = 0$, the optimal control constraint. These three equations are used to compute the gradient of $\mathbf{J}(\mathbf{u}, K)$. The derivation of the adjoint and gradient equations is described in the Appendices.

The solution to an optimization problem that is obtained by a gradient descent algorithm is sensitive to the choice of the initial guess and may be a local minimum of the objective functional rather than the global minimum. To increase the likelihood of obtaining the global minimum, we choose an initial guess for the feature map, represented by $K(\mathbf{x})$, that is guaranteed to include the actual map. This initial guess is that the feature covers the entire area traversed by the swarm during each of its i deployments (in actuality, the feature will occupy a subset of this area). Formally, we define $\gamma_i := [0, 1] \to \mathbb{R}^2$ as the trajectory of the swarm center during the i^{th} deployment and $B_2(\gamma_i(\tau), \delta)$ as a ball with radius δ centered at the point $\gamma_i(\tau)$, and we initially set $K(\mathbf{x}) = 1$ for all $\mathbf{x} \in \left(\cup_{i=1}^{N} B_2(\gamma_i(\tau), \delta)\right) \cap \Omega$, $\tau \in [0, 1]$. We choose δ to be 3 times the standard deviation of the initial Gaussian swarm distribution.

5 Simulated Mapping Scenarios

We developed microscopic and macroscopic models of a robotic swarm for six mapping scenarios, each with a single feature in the domain. The six features varied in position, size, shape, and orientation. We applied the method described in Sect. 4 to reconstruct each feature from the simulated robot data on feature encounter times. For each simulation, we used a swarm of 1000 robots in a normalized domain of size 1 m $\times$ 1 m. The value of k was chosen to be $1/dt$, where dt is the time step of the microscopic model, in order to ensure that robots always switched to the passive state when they encountered the feature boundary. For simplicity, the designated velocity fields $\mathbf{v}_i(t)$ of the robots were each assigned a constant heading. The robots moved at a speed of 0.012 m/s with a diffusion coefficient of $D = 5 \times 10^{-4}$ m^2/s, and each simulation ran for 80 s. The microscopic model was simulated in a 26 $\times$ 26 grid, while the macroscopic model was solved in a finer grid of 51 $\times$ 51 grid cells to account for numerical diffusion. In the optimization procedure, $K(\mathbf{x})$ was bounded between $K_{min} = 0$ and $K_{max} = 1$.

Figure 1 shows snapshots of the active robots in a swarm at various times t during a sample deployment. The robots behave according to the microscopic model and move through a domain that contains a rectangular feature. Robots that have switched to the passive state are not shown. The population of active robots decreases as the robots move eastward and encounter the feature in their path.

Figures 2, 3, 4, 5, 6, 7, 8 and 9 illustrate the results of our mapping procedure for the six scenarios that we investigated. Each figure shows the actual feature, the map of the feature given by the estimated $K(\mathbf{x})$, and the error between these two plots.

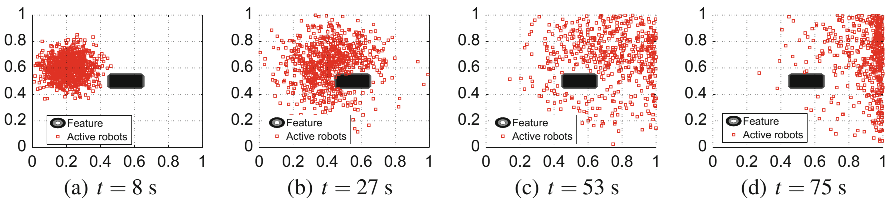


Fig. 1 Snapshots of the simulated swarm moving through a domain with a rectangular feature

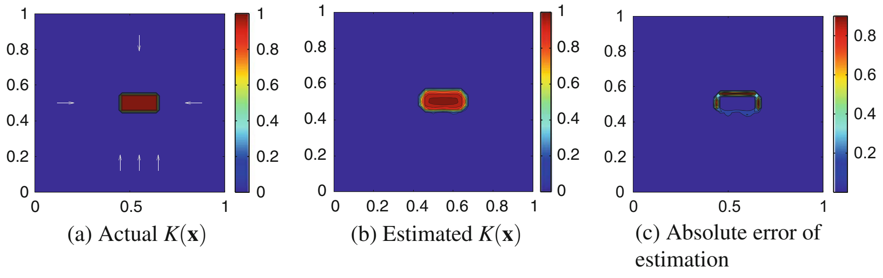


Fig. 2 $K(\mathbf{x})$ estimated from 6 data sets for a domain that contains a *rectangle*. The *white arrows* show the starting locations and directions of the swarm deployments

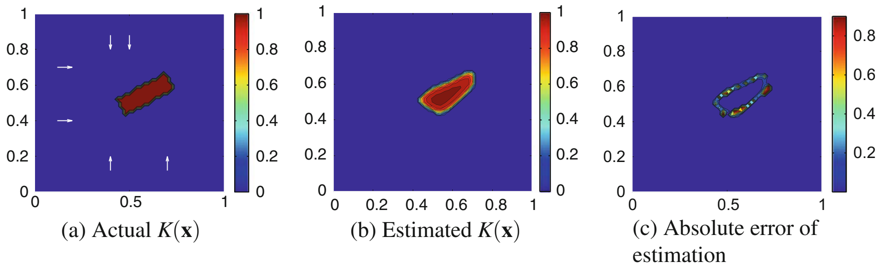


Fig. 3 $K(\mathbf{x})$ estimated from 6 data sets for a domain that contains an inclined *rectangle*

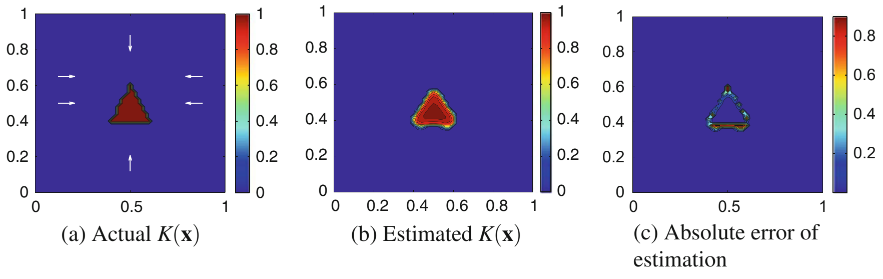
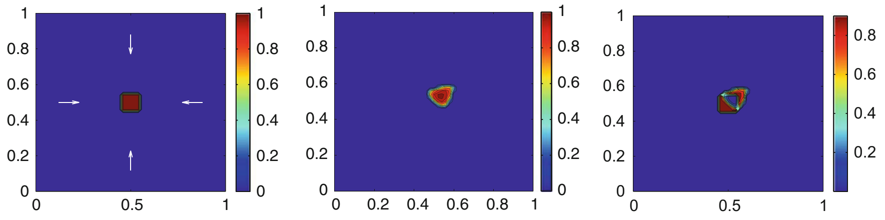
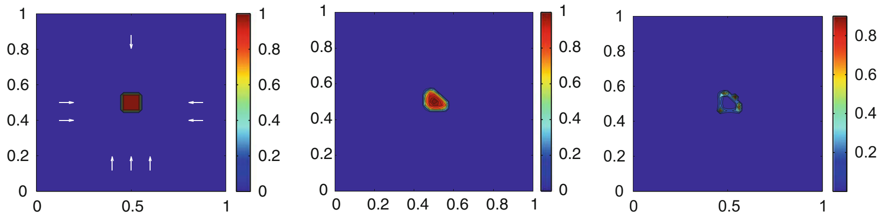


Fig. 4 $K(\mathbf{x})$ estimated from 6 data sets for a domain that contains a *triangle*



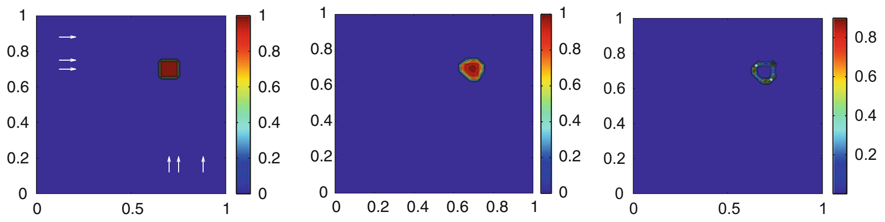
(a) Actual $K(\mathbf{x})$ (b) Estimated $K(\mathbf{x})$ (c) Absolute error of estimation

Fig. 5 $K(\mathbf{x})$ estimated from 4 data sets for a domain that contains a *square* at the center



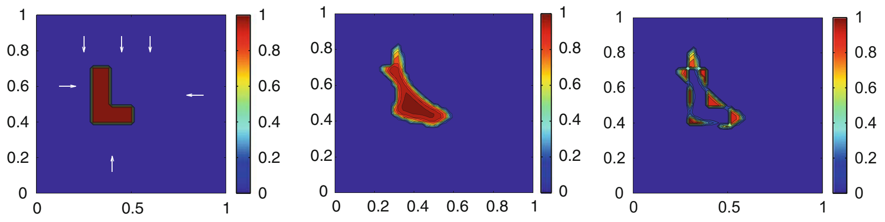
(a) Actual $K(\mathbf{x})$ (b) Estimated $K(\mathbf{x})$ (c) Absolute error of estimation

Fig. 6 $K(\mathbf{x})$ estimated from 8 data sets for a domain that contains a *square* at the center



(a) Actual $K(\mathbf{x})$ (b) Estimated $K(\mathbf{x})$ (c) Absolute error of estimation

Fig. 7 $K(\mathbf{x})$ estimated from 8 data sets for a domain that contains a *square* in the corner



(a) Actual $K(\mathbf{x})$ (b) Estimated $K(\mathbf{x})$ (c) Absolute error of estimation

Fig. 8 $K(\mathbf{x})$ estimated from 6 data sets for a domain that contains a non-convex L-shaped object

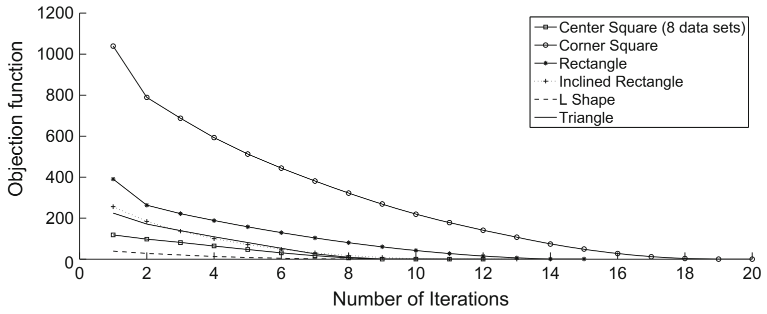


Fig. 9 Objective function value versus number of iterations for the different scenarios examined

In the plots of the actual features, the white arrows indicate the starting points and directions of the swarm center of mass during deployments, each of which yields one data set. Figures 2, 3, and 4 show that we can obtain a fairly accurate map of a rectangle at two different orientations and a triangle using 6 data sets for each scenario. We consider smaller features in the next three figures. From Figs. 5 and 6, we see that the map of a feature increases in accuracy when more non-redundant data sets are used in the optimization procedure. Figure 7 represents a worst-case scenario, in which the map is estimated using data from swarms that start at locations far from the feature, which is in one corner of the domain. The swarms are highly diffused by the time they reach the vicinity of the square; however, 8 data sets yield a relatively accurate map. Lastly, Fig. 8 shows that 6 data sets yield a fairly poor estimate of a non-convex L-shaped feature; we will work further on extending our technique to mapping non-convex shapes. Figure 9 shows that for each scenario considered, the optimal control approach effectively minimizes the objective function by driving it close to zero from its initial value.

6 Conclusion

We have presented a method for mapping an environmental feature using a robotic swarm that exhibits diffusive motion and lacks localization and inter-robot communication. Our approach employs optimal control techniques to reconstruct a spatially varying function that represents the feature of interest. This function is estimated using temporal data on the proportion of active robots, which have not encountered the feature, at each instant of time. Our simulation results indicate that this methodology can accurately reconstruct the feature when the data is obtained from multiple swarm deployments that originate at different locations throughout the domain.

In future work, we would like to extend this approach to more accurately reconstruct non-convex shapes, as well as multiple features in a domain. Our mathematical framework can in principle be used to reconstruct an arbitrary feature geometry, provided that we can design swarm trajectories that yield robot interactions with all facets of the feature. In general, however, it is impossible to identify trajectories a priori that can produce sufficient data for accurate reconstruction. This limitation makes it difficult to map complex feature geometries, as illustrated by Fig. 8. Another factor that contributes to mapping inaccuracies is the decrease in number of active robots during a swarm deployment, which can reduce the correspondence between the density fields of active robots from the macroscopic and microscopic models. This issue could be resolved if the robots perform an obstacle avoidance maneuver upon encountering a feature, staying in the active state rather than entering the passive state. The corresponding macroscopic PDE would need to model this avoidance behavior, which would increase the complexity of computing the gradient of the objective functional. In addition, we plan to implement our mapping approach as the initial step in other swarm strategies, such as collective transport tasks [24] that first require estimating the location and geometry of the payload.

Acknowledgements This work was supported by NSF Awards CMMI-1363499 and CMMI-1436960.

Appendix 1: Mathematical Preliminaries

We study the solution to PDEs in the weak sense, which can be found in the Sobolev space $H^1(\Omega) = \left\{ y \in L^2(\Omega) : \frac{\partial y}{\partial x_1} \in L^2(\Omega), \quad \frac{\partial y}{\partial x_2} \in L^2(\Omega) \right\}$. Here, the spatial derivative is to be understood as a weak derivative defined in the distributional sense. The space is equipped with the common Sobolev space norm, $\|y\|_{H^1(\Omega)} = \sqrt{\left(\|y\|^2_{L^2(\Omega)} + \sum_{i=1}^{2} \left\| \frac{\partial y}{\partial x_i} \right\|^2_{L^2(\Omega)} \right)}$. We also define $V = H^1(\Omega)$, which has the dual space $V^* = H^1(\Omega)^*$.

We consider the general system for Eqs. (3)–(5):

$$\begin{aligned} \frac{\partial u}{\partial t} = Au + \sum_{i=1}^{2} v_i B_i u - K(\mathbf{x})u + f \quad in \;\; L, \\ \mathbf{n} \cdot (D\nabla u - \mathbf{v}u) = g \quad on \;\; \Gamma, \\ u(\mathbf{x}, 0) = u_0, \end{aligned} \tag{9}$$

where A is a formal operator and B_i is an operator defined as $B_i : L^2(0, T; V) \to L^2(0, T; L^2(\Omega))$, $K(\mathbf{x}) \in L^2(\Omega)$, $f \in F = L^2(0, T; L^2(\Omega))$ is the forcing function in the system, $g \in G = L^2(0, T; L^2(\partial\Omega))$, and $u_0 \in L^2(\Omega)$. The variational form of the operator A, called A_g, is defined as $A_g : L^2(0, T; V) \to L^2(0, T; V^*)$. The solution of the system in the weak sense is given by $u \in U = L^2(0, T; V)$ with $u_t \in U^* = L^2(0, T; V^*)$ if it satisfies the equation:

$$\left\langle \frac{\partial u}{\partial t}, \phi \right\rangle_{U^*,U} = \left\langle A_g, \phi \right\rangle_{U^*,U} + \sum_{i=1}^{2} \langle v_i B_i u, \phi \rangle_F - \langle K(\mathbf{x}) u, \phi \rangle_F + \langle f, \phi \rangle_F \quad (10)$$

for all $\phi \in L^2(0, T; V)$. The boundary conditions are equipped with A_g in the variational formulation using Green's theorem. This is essentially the variational form of the Laplacian,

$$\left\langle A_g u, \phi \right\rangle_{U^*,U} = -\langle D\nabla u, \nabla \phi \rangle_{L^2(\Omega)} + \int_{\partial\Omega} (g + \mathbf{n} \cdot \mathbf{v} u)\, \phi dx. \quad (11)$$

In the macroscopic model Eqs. (3)–(5), we define $A = \nabla^2$, $B_i = \frac{\partial}{\partial x_i}$, $f = 0$, and $g = 0$.

Appendix 2: Adjoint Equations

The adjoint equation $\nabla_{\mathbf{u}} \mathscr{L} = 0$ implies that $[\nabla_{u_1} \mathscr{L}, \ldots, \nabla_{u_i} \mathscr{L}, \ldots, \nabla_{u_N} \mathscr{L}] = 0$. From Eq. (8),

$$\begin{aligned} \nabla_{u_i} \mathscr{L} &= \nabla_{u_i} \mathbf{J}(\mathbf{u}, K) + \nabla_{u_i} \sum_{j=1}^{N} \langle p_j, \Psi_j(u_j, K) \rangle \\ &= \nabla_{u_i} \mathbf{J}(u_i, K) + \nabla_{u_i} \langle p_i, \Psi_i(u_i, K) \rangle, \end{aligned} \quad (12)$$

since a term in the sum is a function of u_i only when $i = j$. By Eq. (7),

$$\nabla_{u_i} \mathbf{J}(u_i, K) = \nabla_{u_i} \sum_{j=1}^{N} W_j J_j(u_j) = W_i \nabla_{u_i} J_i(u_i). \quad (13)$$

From Eq. (6),

$$\nabla_{u_i} J_i(u_i) = \nabla_{u_i} \left(\frac{1}{2} \| (Du_i)(t) - g_i(t) \|^2_{L^2([0,T])} \right), \quad (14)$$

where $D := U \to L^2([0, T])$ and $(Du_i)(t) = \int_\Omega u_i(\mathbf{x}, t) d\mathbf{x}$. Then, by the chain rule of differentiation [4, 12], the directional derivative of $J_i(u_i)$, $\nabla_{u_i} J_i(u_i)$, is given by

$$\langle \nabla_{u_i} J_i(u_i), s \rangle_U = \langle (Du_i)(t) - g_i(t), Ds \rangle_{L^2([0,T])} = \langle D^*((Du_i)(t) - g_i(t)), s \rangle_U. \quad (15)$$

Here, $D^* := L^2([0, T]) \to U$ and $(D^* f)(t) = f(t) \cdot \mathbf{1}_\Omega(\mathbf{x})$, where $f(t) \in L^2([0, T])$ and $\mathbf{1}_\Omega$ is the indicator function of $\Omega \subset \mathbb{R}^2$. We can show that $\langle Dy, f \rangle = \langle y, D^* f \rangle \; \forall y \in U, \; f \in L^2([0, T])$. Therefore,

$$\nabla_{u_i} J_i(u_i) = D^*((Du_i)(t) - g_i(t)). \tag{16}$$

By definition,

$$\langle p_i, \nabla_{u_i} \Psi_i(u_i, K)s\rangle = \langle \nabla_{u_i} \Psi_i(u_i, K)^* p_i, s\rangle \;\; \forall s \in U, \tag{17}$$

where $\nabla_{u_i} \Psi_i(u_i, K)^*$ is the adjoint operator of $\nabla_{u_i} \Psi_i(u_i, K)$ corresponding to the inner product of the Hilbert space. Now, by taking the directional derivative of $\Psi_i(u_i, K)$ at u_i in the direction of s, we obtain

$$\nabla_{u_i} \Psi_i(u_i, K)s = \frac{\partial s}{\partial t} - (\nabla \cdot (D\nabla s - \mathbf{v}_i(t)s) - kK(\mathbf{x})s). \tag{18}$$

Substituting Eq. (18) into Eq. (17) yields

$$\langle p_i, \nabla_{u_i} \Psi_i(u_i, K)s\rangle = \int_0^T \langle p_i, \frac{\partial s}{\partial t}\rangle_{L^2(\Omega)} - \langle p_i, D\nabla^2 s\rangle + \langle p_i, \nabla \cdot \mathbf{v}_i(t)s\rangle + \langle p_i, kK(\mathbf{x})s\rangle.$$

Using integration by parts on the integral term in the equation above, we get

$$\int_0^T \langle p_i, \frac{\partial s}{\partial t}\rangle_{L^2(\Omega)} = \langle p_i(T), s(T)\rangle - \langle p_i(0), s(0)\rangle - \int_0^T \langle s, \frac{\partial p_i}{\partial t}\rangle_{L^2(\Omega)}.$$

As this is true for all $s \in U$, we could choose the s with $s(0) = 0$ and construct $p_i(T)$ such that $\int_0^T \langle p_i, \frac{\partial s}{\partial t}\rangle_{L^2(\Omega)} = \int_0^T \langle -\frac{\partial p_i}{\partial t}, s\rangle_{L^2(\Omega)}$. Thus, we choose the final condition of the adjoint equation as $p_i(T) = 0$. We now make use of the following lemma:

Lemma 1 *Let L and L^* be operators defined by $L : L^2(0, T; V) \to L^2(0, T; V^*)$ and $L^* : L^2(0, T; V) \to L^2(0, T; V^*)$, respectively. The variational form of L is:*

$$\langle Lu, \phi\rangle_{V^*,V} = -\langle D\nabla u, \nabla\phi\rangle_{L^2(\Omega)} - \langle \mathbf{v} \cdot \nabla u, \phi\rangle_{L^2(\Omega)} + \int_{\partial\Omega} \mathbf{n} \cdot (\mathbf{v}u\phi)dx$$

$\forall\phi \in V$. *Also, by Lagrange's identity,* $\langle Lu, p\rangle_{V^*,V} = \langle u, L^* p\rangle_{V,V^*}$ $\forall u, p \in L^2(0, T; V)$. *We use the zero-flux boundary condition in Eq. (4) to compute the variational form of the operator* L^* *to be* $\langle L^* p, \phi\rangle_{V^*,V} = -\langle D\nabla p, \nabla\phi\rangle_{L^2(\Omega)} + \langle \mathbf{v} \cdot \nabla p, \phi\rangle_{L^2(\Omega)}$ $\forall p \in L^2(0, T; V)$ *and* $\forall\phi \in V$.

Using the variational form of the Laplacian as in Eq. (11) and applying Lemma 1 and integration by parts, we can show that $-\langle p_i, D\nabla^2 s\rangle + \langle p_i, \nabla \cdot \mathbf{v}_i(t)s\rangle$ can be transformed into $-\langle D\nabla^2 p_i, s\rangle - \langle \nabla \cdot \mathbf{v}_i(t)p_i, s\rangle$ with the boundary condition $\mathbf{n} \cdot \nabla p_i = 0$. Finally, we observe that $\langle p_i, K(\mathbf{x})s\rangle = \langle p_i K(\mathbf{x}), s\rangle$. By combining these results with Eqs. (12), (15), and (17), we obtain

$$\langle \nabla_{u_i} J_i(u_i), s\rangle + \langle -\frac{\partial p_i}{\partial t} - D\nabla^2 p_i - \nabla \cdot \mathbf{v}_i(t)p_i + p_i kK(\mathbf{x}), s\rangle = 0.$$

Thus, the set of adjoint equations for the system defined by the i^{th} set of constraints, $\Psi_i(u_i, K)$, with respect to the objective functional, $\mathbf{J}$, is given by

$$-\frac{\partial p_i}{\partial t} = \nabla \cdot (D\nabla p_i + \mathbf{v}_i(t)p_i) - p_i k K(\mathbf{x}) - \nabla_{u_i} J_i(u_i) \quad in \;\; L \tag{19}$$

with the Neumann boundary conditions

$$\mathbf{n} \cdot \nabla p_i = 0 \quad on \;\; \Gamma, \quad p_i(T) = 0, \quad i = 1, \ldots, N. \tag{20}$$

Here, Eq. (19) with Eq. (20) has a solution in the weak sense.

Appendix 3: Gradient Equation

Using a similar analysis to the one in Appendix 2, we find that $\nabla_K \mathscr{L}$ reduces to

$$\nabla_K \mathscr{L} = \nabla_K \mathbf{J}(\mathbf{u}, K) + \sum_{i=1}^{N} \nabla_K \langle p_i, \Psi_i(u_i, K)\rangle. \tag{21}$$

From Eq. (7), we can derive the following expressions:

$$\nabla_K \mathbf{J}(\mathbf{u}, K) = \nabla_K \frac{\lambda}{2} \|K(\mathbf{x})\|^2_{L^2(\Omega)}, \quad \langle \nabla_K \mathbf{J}(\mathbf{u}, K), s\rangle = \langle \lambda K(\mathbf{x}), s\rangle. \tag{22}$$

As in Appendix 2, we could express $\langle p_i, \nabla_K \Psi_i(u_i, K)s\rangle$ as $\langle \nabla_K \Psi_i(u_i, K)^* p_i, s\rangle \; \forall s \in L^2(\Omega)$, where $\nabla_K \Psi_i(u_i, K)^*$ is the adjoint operator of $\nabla_K \Psi_i(u_i, K)$ corresponding to the inner product of the Hilbert space. Now, by taking the directional derivative of $\Psi_i(u_i, K)$ at K in the direction of s, we find that $\nabla_K \Psi_i(u_i, K)s = k u_i s$. Therefore, with further simplification, we can show that

$$\langle \nabla_K \Psi_i(u_i, K)^* p_i, s\rangle = \langle (\Xi(k u_i p_i))(\mathbf{x}), s\rangle_{L^2(\Omega)}, \tag{23}$$

where $\Xi := L^2(0, T; \Omega) \to L^2(\Omega)$ and $(\Xi f)(\mathbf{x}) = \int_0^T f dt$ for all $f \in L^2([0, T]; \Omega)$ and $\mathbf{x} \in \Omega$. By combining Eqs. (21)–(23), we formulate the objective functional derivative as

$$\mathbf{J}' = \sum_{i=1}^{N} (\Xi(k u_i p_i))(\mathbf{x}) + \lambda K(\mathbf{x}). \tag{24}$$

Thus, the computation of $\mathbf{J}'$ requires u_i and p_i, which can be obtained by solving $\Psi_i(u_i, K)$ forward and solving Eqs. (19), (20) backward.

References

1. Ammari, H.: An Introduction to Mathematics of Emerging Biomedical Imaging, vol. 62. Springer, Berlin (2008)
2. Belmiloudi, A.: Stabilization, Optimal and Robust Control: Theory and Applications in Biological and Physical Sciences. Springer, London (2008)
3. Biswas, R., Limketkai, B., Sanner, S., Thrun, S.: Towards object mapping in dynamic environments with mobile robots. In: International Conference on Intelligent Robots and Systems (IROS) (2002)
4. Borzì, A., Schulz, V.: Computational optimization of systems governed by partial differential equations. In: Society for Industrial and Applied Mathematics (SIAM), Philadelphia (2012)
5. Correll, N., Hamann, H.: Probabilistic modeling of swarming systems. In: J Kacprzyk WP (ed) Springer Handbook of Computational Intelligence, pp 1423–1431. Springer, Heidelberg (2015)
6. Dirafzoon, A., Lobaton, E.: Topological mapping of unknown environments using an unlocalized robotic swarm. In: International Conference on Intelligent Robots and Systems (IROS) (2013)
7. Dirafzoon, A., Betthauser, J., Schornick, J., Benavides, D., Lobaton, E.: Mapping of unknown environments using minimal sensing from a stochastic swarm. In: International Conference on Intelligent Robots and Systems (IROS) (2014)
8. Elamvazhuthi, K.: A variational approach to planning, allocation and mapping in robot swarms using infinite dimensional models. Master's thesis, Arizona State University (2014)
9. Elamvazhuthi, K., Berman, S.: Optimal control of stochastic coverage strategies for robotic swarms. In: International Conference on Robotics and Automation (ICRA) (2015)
10. Fattorini, H.O.: Infinite Dimensional Optimization and Control Theory, vol 54. Cambridge University Press, Cambridge (1999)
11. Gardiner, C.W.: Stochastic Methods: A Handbook for the Natural and Social Sciences, 4th edn. Springer, Evanston, IL, USA (2009)
12. Hinze, M., Pinnau, R., Ulbrich, M., Ulbrich, S.: Optimization with PDE Constraints, vol. 23. Springer, Netherlands (2009)
13. Horning, M., Lin, M., Siddarth, S., Zou, S., Haberland, M., Yin, K., Bertozzi, A.L.: Compressed sensing environmental mapping by an autonomous robot. In: Proceedings of the Second International Workshop on Robotic Sensor Networks. Seattle, WA (2015)
14. Kirsch, A.: An Introduction to the Mathematical Theory of Inverse Problems, 2nd edn. vol 120. Springer, New York (2011)
15. Kuipers, B., Byun, Y.T.: A robot exploration and mapping strategy based on a semantic hierarchy of spatial representations. Int. J. Robot. Auton. Syst. **8**(1–2), 47–63 (1991)
16. Liu, S., Mohta, K., Shen, S., Kumar, V.: Towards collaborative mapping and exploration using multiple micro aerial robots. In: International Symposium on Experimental Robotics (ISER) (2014)
17. Robb, R.A.: Biomedical Imaging, Visualization, and Analysis. Wiley Inc, New York, NY, USA (1999)
18. Robertson, P., Angermann, M., Krach, B.: Simultaneous localization and mapping for pedestrians using only foot-mounted inertial sensors. In: Proceedings of the 11th International Conference on Ubiquitous Computing (Ubicomp) (2009)
19. Thrun, S.: A probabilistic online mapping algorithm for teams of mobile robots. Int. J. Robot. Res. **20**(5), 335–363 (2001)
20. Thrun, S., Bücken, A.: Integrating grid-based and topological maps for mobile robot navigation. In: Proceedings of the AAAI 13th National Conference on Artificial Intelligence (1996)
21. Tong, S., Fine, E.J., Lin, Y., Cradick, T.J., Bao, G.: Nanomedicine: tiny particles and machines give huge gains. Ann. Biomed. Eng. **42**(2), 243–259 (2014)
22. Tröltzsch, F.: Optimal Control of Partial Differential Equations: Theory, Methods, and Applications, vol. 112. American Mathematical Society, Providence, RI, USA (2010)

23. Tuchin, V.V.: Tissue Optics, Light Scattering Methods and Instruments for Medical Diagnosis, 3rd edn. vol. PM254. Spie Press Book (2015)
24. Wilson, S., Pavlic, T.P., Kumar, G.P., Buffin, A., Pratt, S.C., Berman, S.: Design of ant-inspired stochastic control policies for collective transport by robotic swarms. Swarm Intell. **8**(4), 303–327 (2014)

An Effective Algorithmic Framework for Near Optimal Multi-robot Path Planning

Jingjin Yu and Daniela Rus

1 Introduction

We study the problem of planning collision-free paths for multiple labeled disc robots operating in two-dimensional, multiply-connected, continuous environments (i.e., environments with holes). The *primary goal* of this work is to develop a practical, extensible framework toward the efficient resolution of multi-robot path planning (MPP) problems, in which the robots are densely packed, while simultaneously seeking to minimize *globally the task completion time*. The framework is composed of two key algorithmic components, executed in an sequential order. Using the example illustrated in Fig. 1a, first, we compute the configuration space for a single robot, over which an optimal lattice structure is overlaid (Fig. 1b). Using the lattice structure as a roadmap, each start (resp., goal) location is assigned to a nearby node of the roadmap as its unique discrete start (resp., goal) node, which translates the continuous problem into a discrete one (Fig. 1c). Then, a state-of-the-art discrete planning algorithm is applied to solve the roadmap-based problem near-optimally (Fig. 1d). Through the tight composition of these two algorithmic components, our framework proves to be highly effective in a variety of settings, pushing the boundaries on optimal multi-robot path planning to new grounds in terms of the number of robots supported and the allowed robot density.

Related work. MPP finds applications in a wide spectrum of domains such as navigation [1, 25], manufacturing and assembly [13], warehouse automation [40], computer video games [26], and microfluidics [8]. Given the important role it holds in robotics-related applications, MPP problems has received considerable attention in robotics

J. Yu (✉)
Computer Science, Rutgers University, New Brunswick, NJ, USA
e-mail: jingjin.yu@rutgers.edu

D. Rus
CSAIL, Massachusetts Institute of Technology, Cambridge, MA, USA
e-mail: rus@csail.mit.edu

A. Bicchi and W. Burgard (eds.), *Robotics Research*, Springer Proceedings in Advanced Robotics 2, DOI 10.1007/978-3-319-51532-8_30

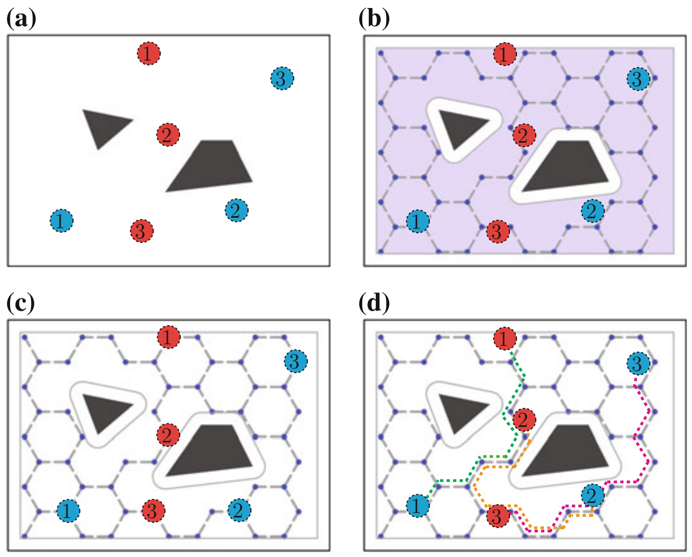


Fig. 1 An illustrative example of our algorithmic framework. **a** A problem instance with three disc robots. The start and goal locations are indicated by the *blue* and *red* labeled discs, respectively. **b** The configuration space (*shaded area*) for a single robot and the fitted hexagonal lattice. The *blue circles* are the start positions, and the *red circles* are the goal positions. **c** The discrete abstraction of the original problem. **d** Solution to the original continuous problem

research with dedicated study on the subject dating back at least three decades [24], in which a centralized approach is taken that considers all robots as a single entity in a high dimensional configuration space. Because the search space in such problems grows exponentially as the number of robots increases linearly, a centralized approach [24], although complete, would be extremely inefficient in practice. As such, most ensuing research take the approach of decomposing the problem. One way to do this is by assigning priorities to the robots so that robots with higher priority take precedence over robots with lower priority [3, 5]. Another often adopted partitioning method is to plan a path for each robot separately without considering robot-robot interaction. The paths are then *coordinated* to yield collision free paths [2, 19]. Following these initial efforts, the decomposition scheme is further exploited and improved [7, 17, 21, 31, 34, 35]. Many of the mentioned works also consider optimality in some form. We emphasize that, since finding feasible solution for MPP is already PSPACE-hard [10], i.e., no polynomial-time complete algorithm may even exist for such problems unless P = PSPACE, computing globally near-optimal solution for a large number of robots is extremely challenging.

Recent years have witnessed a great many new approaches being proposed for solving MPP. One such method, reciprocal velocity obstacles [33, 36], which can be traced back to [11], explicitly looks at velocity-time space for coordinating robot motions. In [8], mixed integer programming (MIP) models are employed to encode the interactions between the robots. A method based on network-flow is explored in [37]. In [20], similar to our framework upon a first look, an A^*-based search is performed over a discrete roadmap abstracted from the continuous environment. However, the authors addressed a much narrower class of problems for which they can bound the computation cost but cannot guarantee the solution optimality. It is also unclear how the complex geometric problem of efficiently computing a discrete roadmap from the continuous environment is resolved in the paper. In [28], discrete-RRT (d-RRT) is proposed for the efficient search of multi-robot roadmaps. Lastly, as a special case of MPP in continuous domains, efficient algorithms are proposed [29, 32] for interchangeable robots (i.e., in the end, the only requirement is that each goal location is occupied by an arbitrary robot). At the same time, discrete (e.g., graph-based) MPP has also been a subject of active investigation. This line of research originates from the mathematical study of the 15-puzzle and related *pebble motion* problems [14, 39]. Since then, many heuristics augmenting the A^* algorithm have been proposed for finding optimal solution, e.g., [23, 30, 38], to name a few. These heuristics essentially explore the same decoupling idea used in the continuous case to trim down the search space. A method based on network-flow also exists here [42]. Some of these discrete solutions, such as [14], have helped solving continuous problems [15, 27].

Contribution. Our work brings two contributions toward solving MPP effectively and optimally. First, we introduce a two-phase framework that allows any roadmap building (i.e., discretization) method to be combined with any suitable discrete MPP algorithm for solving continuous MPP problems. The framework achieves this by imposing a partial collision avoidance constraint during the roadmap building phase while preserving path near-optimality. Second, we deliver a practical integrated algorithmic implementation of the two-phase framework for computing near optimal paths for a large number of robots. We accomplish this by combining *(i)* a fast algorithm for superimposing dense regular lattice structures over a bounded two-dimensional environment with holes and *(ii)* an integer linear programming (ILP) based algorithm for computing near-time-optimal solutions to discrete MPP [41]. To the best of our knowledge, we present the first such algorithm that can quickly plan near optimal, continuous paths for hundreds of robots densely populated in multiply-connected environments.[1]

Paper organization. The rest of the paper is organized as follows. We formulate the MPP problem in Sect. 2. In Sect. 3, we describe the overall algorithmic framework architecture and the first component of the framework on roadmap-based problem construction. In Sect. 4, we describe how the second component of the framework may be realized. In Sect. 5, we demonstrate the effectiveness of our framework over a variety of environments. We hold an extensive discussion and conclude in Sect. 6.[2]

[1]Warehousing systems from Kiva Systems [40] can work effectively with hundreds of robots. However, these robots essentially live on a grid within a structured environment.

[2]Due to limited space, detailed description of the ILP algorithm (Sect. 4) and larger versions of some figures are included in the online material available at http://arxiv.org/abs/1505.00200. An

2 Problem Statement

Let $\mathscr{W}$ denote a bounded, open, multiply-connected (i.e., with holes), two-dimensional region. We assume that the boundary and obstacles of $\mathscr{W}$ can be approximated using polygons with an overall complexity of m (i.e., there are a total of m edges). There are n unit disc robots residing in $\mathscr{W}$. These robots are assumed be omnidirectional with a velocity v satisfying $|v| \in [0, 1]$. Let $\mathscr{C}_f$ denote the free configuration space for a single robot (the shaded area in Fig. 1b). The centers of the n robots are initially located at $S = \{s_1, \ldots, s_n\} \subset \mathscr{C}_f$, with goals $G = \{g_1, \ldots, g_n\} \subset \mathscr{C}_f$. For all $1 \leq i \leq n$, a robot initially located at s_i must be moved to g_i.

In addition to planning collision-free paths, we are interested in optimizing path quality. Our particular focus in this paper is minimizing the *global task completion time*, also commonly known as *makespan*.[3] Let $P = \{p_1, \ldots, p_n\}$ denote a feasible path set with each p_i a continuous function, defined as

$$p_i : [0, t_f] \to C_f,\ p_i(0) = s_i,\ p_i(t_f) = g_i.$$

The *makespan* objective seeks solutions that minimize t_f. In other words, let $\mathscr{P}$ denote the set of all solution path sets, the task is to find a path set with t_f close to

$$t_{min} := \min_{P \in \mathscr{P}} t_f(P). \tag{1}$$

We emphasize that the aim of this work is a method for quickly solving "typical" problem instances with many robots and high robot density (i.e., the ratio between robot footprint and the free space is high) with optimality assurance. By *typical*, we mean that: *(i)* the start location and goal locations are reasonably separated, *(ii)* a start or goal location is not too close to static obstacles in the environment, and *(iii)* there are no narrow passages in the environment that cause the discretized roadmap structure to have poor connectivity. More formally, we assume that assumptions *(i)* and *(ii)*, respectively, take the forms[4]

$$\forall 1 \leq i, j \leq n, \quad |s_i - s_j| \geq 4, \ \ |g_i - g_j| \geq 4 \tag{2}$$

and

$$\forall p \in \{S \cup G\}, \quad |p - q| \leq \sqrt{5} \Rightarrow q \in \mathscr{W}. \tag{3}$$

(Footnote 2 continued)
accompanying video demonstrating our algorithm and software developed in this paper are available from the corresponding author's website.

[3]Note that our algorithmic framework also applies to other time- and distance-based optimality objectives through the use of an appropriate discrete planning algorithm.

[4]Equations (2) and (3) are unit-less given the unit disc robot assumption. If the robots have radius r, the right side of the inequalities from (2) and (3) should be scaled by a multiplicative factor of r.

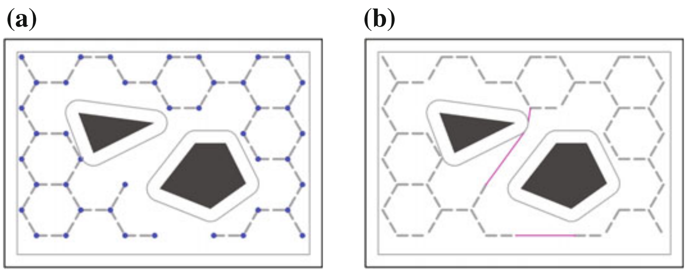


Fig. 2 **a** An environment with a discretization that does not capture its original topology. **b** The roadmap after restoring connectivity (the operations are performed automatically from our code), which then captures the topology of the original environment

For *(iii)*, the discretized roadmap should capture the topology of the continuous environment well. To be more concrete, see Fig. 2a. In this environment, there are two holes. The lattice graph, after contraction of faces that do not contain any obstacles, does not have any holes. We expect the discrete roadmap to be connected and have number of holes (after face contraction) equal to the number of holes of the continuous environment (e.g., Fig. 1d).
Remark. We provide these assumptions only to suggest situations in which our framework is expected to perform well. In our evaluation, these assumptions are not enforced. We in fact greatly relax (2) (from 4 to 2.5) and do not enforce (3) at all. We also give an efficient subroutine for restoring connectivity when assumption *(iii)* is not satisfied. For example, the routine, when applied to the example in Fig. 2a, yields the result in Fig. 2b, which is a screen capture from our program. We also emphasize that, given that optimal MPP is an extremely challenging task computationally [10] and our focus on method effectiveness, we do not consider the problem from the angle of solution completeness.

3 Algorithmic Framework Architecture and Roadmap-Based Discrete Problem Construction

We solve the proposed problem using an algorithmic framework with two algorithmic components–discretization of the continuous problem followed by resolution of the roadmap-based problem. The overall framework contains four sequential procedures:

(i) select and overlay a regular lattice structure over the configuration space,
(ii) restore environment connectivity lost in the discretization process,
(iii) snap start and goal locations to roadmap nodes to create a discrete problem on the roadmap, and
(iv) solve the discrete MPP problem optimally or near-optimally.

We note that, when compared with motion planning methods such as PRM [12] and RRT [16], our framework, looking somewhat similar on the surface, is in fact rather different. In methods like PRM and RRT, the discretization deals with the configuration space encompassing all degrees of freedom of the target system. Our approach, on the other hand, performs a careful, mostly uniform discretization of the configuration space for a single robot with two degrees of freedom. In doing so, we trade probabilistic completeness for the faster computation of near-optimal solutions. In the rest of this section, we describe the first key component of our algorithmic framework–the construction of the roadmap-based discrete problem, which subsumes the first three algorithmic procedures of the overall framework.

3.1 Lattice Selection and Imposition

Appropriate lattice structure selection In selecting the appropriate lattice structure, we aim to allow the packing of more robots simultaneously on the resulting roadmap and obtain the structure fast. Clearly, if an insufficient number of nodes exists in the roadmap, the resulting discrete problem can be crowded with robots, which is difficult to solve and may not even have a solution. On the other hand, to allow a clean separation between the roadmap building phase and the discrete planning phase of the framework, the nodes cannot be too close to each other, e.g., two robots occupying two different nodes should not be in collision. Moreover, it is desirable that two robots moving on different edges in parallel will not collide with each other.

Considering all these factors together, we resort to adopting uniform tilings of the plane [22]. A uniform tiling of the plane is a regular network structure that can be repeated infinitely to cover the entire two-dimensional plane. Due to the regularity of uniform tilings, it is computationally easy to overlay a tiling pattern over $\mathscr{C}_f$. Choosing such a tiling then relieves us from selecting each node for the roadmap individually. Over the 11 uniform tilings[5] of the plane [22], we computed the density of robots supported by each. To allow concurrent moves of robots on nearby edges, take square tiling as an example, a square must have a side length of $4/\sqrt{2}$ to avoid potential collision incurred by such moves (see, e.g., Fig. 3a). Indeed, it is straightforward to show that the closest inter-robot distance is reached when two robots are in the middle of two edges connecting to the same node. For hexagonal tilings, this results in a minimum side length of $4/\sqrt{3}$ (Fig. 3b).

After obtaining the required side length parameters for all 11 tilings, the maximum robot density allowed by these tilings can then be computed. We compute the density by assuming that all nodes of the regular tiling patterns are occupied by robots and compute the ratio between the area occupied by robots and the free space when it is unoccupied. For an infinite lattice with no obstacles, the hexagonal tiling is the best

[5]These tilings are: triangular, trihexagonal, square, elongated triangular, hexagonal, truncated square, truncated trihexagonal, truncated hexagonal, snub square, rhombitrihexagonal, snub hexagonal.

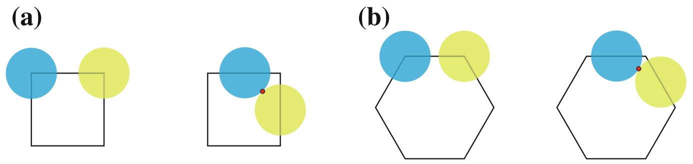


Fig. 3 Minimum distance between robots. To ensure no collision when executing a discrete plan, the distance between two lattice nodes must be $4/\sqrt{2} + \varepsilon$ for *square tilings* (**a**) and $4/\sqrt{3} + \varepsilon$ for *hexagonal tilings* (**b**). At exactly $4/\sqrt{2}$ (resp. $4/\sqrt{3}$) the robots will touch when reaching the midpoint of the edge. The contact point is shown as a *red* doc in both figures

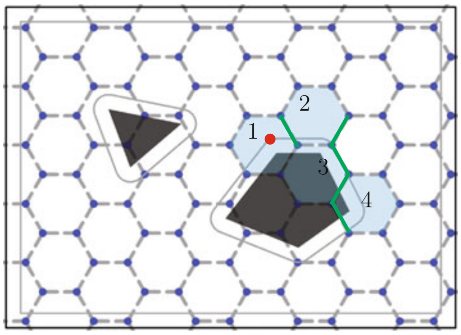


Fig. 4 Efficient computation of the hexagonal lattice that falls inside $\mathcal{C}_f$

with about 45% density, followed by the square tiling with roughly 39% density. Triangular tilings have a density of only 23%. This leads us to choose hexagonal lattices as the base structure of the discrete roadmap.

Imposing the lattice structure After deciding on the lattice structure, we need a procedure for imposing the structure on $\mathcal{C}_f$. Essentially, every edge must be checked to determine whether it is entirely contained in the free configuration space $\mathcal{C}_f$. Note that if this is performed naively, i.e., performing collision checking of each edge with all obstacles, the overall complexity is on the order of $O(mA)$, in which m is the complexity of the workspace and A is the area contained in the outer boundary. The naive approach quickly becomes time-consuming as either m or A grows.

To complete this step efficiently, we start by making an arbitrary alignment between a sufficiently large piece of the infinite hexagonal lattice and the continuous environment (Fig. 4). Then, we look at one C-space obstacle (including the outer boundary) at a time. For each obstacle, we pick an arbitrary vertex on the boundary (red dot in Fig. 4) and locate the hexagon from the lattice it belongs to (in case of the example in Fig. 4, the shaded hexagonal with the label "1"). We then follow the obstacle boundary and find all (green) edges of the lattice that intersect the boundary. The edges found this way do not belong to $\mathcal{C}_f$ and the final discrete graph structure; moreover, they partition the lattice into pieces that are either completely inside $\mathcal{C}_f$ or completely outside $\mathcal{C}_f$. This allows us to efficiently check whether the rest of the lattice edges belong to $\mathcal{C}_f$. To do so, we start with a vertex that is within $\mathcal{C}_f$ that also belongs to one of these green edges and perform a breath first search over

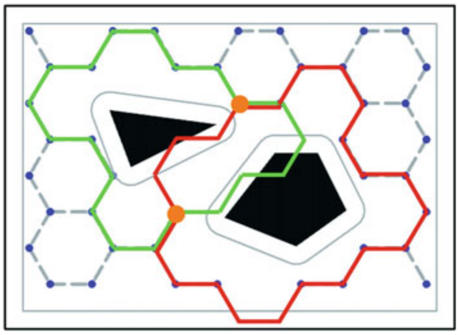


Fig. 5 Smallest cycles fully surrounding the two $\mathscr{C}_f$ obstacles

the lattice structure, now with all the green edges deleted. All edges found this way must be long to $\mathscr{C}_f$. We repeat this until all vertices of the lattice that fall inside $\mathscr{C}_f$ are exhausted. Note that this BFS is a discrete search without performing geometric computation over real numbers, which can be done much faster than edge intersection checks. In the end, we obtain an output sensitive algorithm that typically takes time between $\Theta(\sqrt{A})$ and $\Theta(A)$, depending the total length of obstacle boundaries. In practice, using the said method, the computation time used by this step is trivial in comparison to the time it takes to do the discrete planning.

Restore Configuration Space Connectivity We now address how we may ensure that the topology of $\mathscr{C}_f$ is preserved in the discrete roadmap. Essentially, we must locate places where connectivity in the continuous environment is lost. We illustrate our algorithmic solution for doing so using an example. For the problem given in Fig. 3a, for each C-space obstacle, it is straightforward to obtain the smallest cycle on the lattice enclosing the obstacle (e.g., the green and red cycles in Fig. 5). Then, for each pair of obstacles, we check whether the corresponding enclosing cycles share non-trivial interior and if so, locate a minimum segment on the overlapping section (e.g., the red segments between the two orange nodes in Fig. 5). Using *visibility graph* [18], we may then restore the lost connectivity and obtain the roadmap shown in Fig. 3b. Most of the computation time in this step is spent on computing the visibility graph itself, which takes time $O(m \log m + E)$ [6], with m being the complexity of the environment and E being the number of edges in the resulting visibility graph.

Remark. In the process of restoring connectivity, it is possible that the resulting roadmap cannot guarantee that simultaneous movements of disc robots are collision-free. Without getting into details, we mention that this issue can be fully addressed by sacrificing some time optimality.

We also note that the preservation of the connectivity or topology of the continuous environment can be crucially important. A better connected environment has a more diverse set of candidate paths, making the resulting problem easier to solve. Perhaps more importantly, the preservation of the connectivity of $\mathscr{C}_f$ is essential to preserving path optimality. For a roadmap built from an overlaid square lattice, given a shortest path $p \subset \mathscr{C}_f$ between two points, due to the *strong equivalence* between the Euclidean metric and the Manhattan metric, the shortest path p and the corre-

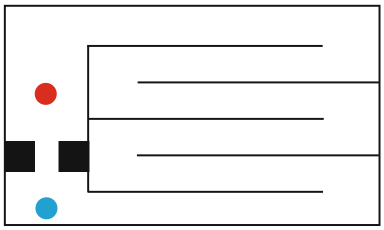


Fig. 6 Suppose that the start and goal locations are at the *center* of the *blue* and the *red* discs, respectively. If the robot does not find the narrow passage on the *left*, it then needs to travel through a winding path on the *right*. By extending the width of the environment, we can make the winding path arbitrarily long when compared to the shortest path

sponding shortest path p' on the square lattice-based roadmap are within a constant factor multiple of each other for any reasonably long path p (that is, $length(p) \ll 1$ does not hold). The same argument applies to the roadmap-based hexagonal lattices. Without obstacles, the ratio $length(p')/length(p)$ over a long path p is bounded by $\sqrt{2}$ for square lattices and roughly the same for hexagonal lattices. The ratio is largely the same when obstacles are present. On the other hand, if the connectivity of $\mathcal{C}_f$ is not preserved, then it becomes possible that $length(p')/length(p)$ is arbitrarily large. An example is given in Fig. 6.

Once we establish that the roadmap preserves the near-optimality on path length, the same applies to time optimality. Given the preservation of near-optimality of individual paths, it does not directly imply that an optimal solution to the abstracted discrete problem also preserves optimality with respect to the original continuous problem, in terms of time or distance. However, our computational experiments show that this is generally the case when $\mathcal{C}_f$ has good connectivity.

3.2 Snapping Start and Goal Locations to Roadmap Nodes

After the full roadmap is built, each start or goal location in $S \cup G$ must be associated with a nearby roadmap node. We call this process snapping. For the snapping step, for each $s_i \in S$, we simply associate s_i with the closest roadmap node that s_i can reach without colliding with another $s_j \in S$. The same process is performed for all $g_i \in G$ With the separation assumptions (2) and (3), this is almost always possible. In particular, (2) implies that each hexagon from the lattice contains (roughly) at most one start and one goal location. Therefore, the number of nodes on the roadmap is at least twice the number of robots. In rare cases when conflicts do happen, we may apply the rearrangement algorithms (e.g., [29]) to perform the snapping step without incurring much penalty on time optimality. The completeness of this step is guaranteed by (2) and (3).

With the snapping process complete, a discrete abstraction of the original continuous problem is obtained. For our example, this leads to the scenario captured in Fig. 1c. If we are not interested in optimality, the discrete problem may be attempted using a non-optimal but polynomial time algorithm [14, 44]. As stated in the individual subsections, the computation required in this section can be carried out using low-degree polynomial time algorithms. The relative time used for this portion is trivial as compared to the time required for solving the roadmap-based discrete problem.

4 Fast, Near-Optimal Discrete Path Planning

After a high quality roadmap is obtained with near-optimality guarantees on time and distance (e.g., an optimality-preserving reduction from continuous space to discrete space), one may then freely choose an algorithm for finding solutions to the discrete abstraction (Fig. 1c in our example). Whereas an arbitrary number of globally optimal objectives can be conjured, four objectives are perhaps most natural. These four objectives minimize the maximum or the total *arrival time* or *travel distance*. Viewing from the angle of service provider (e.g., delivery drones) and end user (e.g., customers), minimizing the total distance or time allows the service provider to minimize energy cost or overall vehicle fleet usage. On the other hand, minimizing the maximum time or distance promises a more uniform service quality among customers. If minimizing the total arrival time or the total distance is the goal, then discrete search methods such as ID [30] can be applied. Here, we focus on the minimum *makespan* (i.e., maximum arrival time or task completion time). We describe an effective method for minimizing the makespan [41, 42], which is also a good proxy to minimizing the maximum travel distance. The method is an ILP-based one with an optimal baseline algorithm, augmented with near-optimal heuristics to improve the computational performance.
The baseline, ILP model-based algorithm We first describe how an ILP model is obtained [42]. The key idea is to perform *time expansion* over the discrete (spatial) roadmap and then build the ILP model over the resulting directed *space-time* graph. This step essentially sequentially chains some T copies of the spatial roadmap together. Locally, for a hexagonal roadmap, the space-time graph has the structure given in Fig. 7. Now, if the discrete MPP allows a solution within T time steps, then there is a corresponding solution on the space-time graph in the form of *n vertex disjoint paths*. An ILP model can then be readily build to find these vertex disjoint

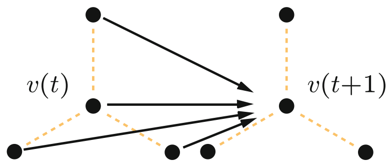


Fig. 7 In a single time expansion step, a node's neighbors (including the node itself) at time step t are connected to the node at time step $t + 1$

paths. Each solution found on the space-time graph can be easily mapped back to yield a feasible solution to the original MPP problem. To ensure optimality, a conservative initial estimate of T, the number of steps for doing time expansion, is used. This T is then gradually increased until a first feasible solution is found, which is also a minimum makespan solution.

k-way split heuristic As finding minimum makespan solutions to discrete MPP is NP-hard [43], we observe the time in solving the corresponding ILP models grows exponentially as the size of the model grows. This lead us to a heuristic that breaks a large ILP model into multiple small ones along the *time line*. If the problem is broken in k pieces, we call it a k-way split. Using 2-way split as an example, first, individual paths for the robots are computed. Then the mid-point of these paths are used to divide the discrete MPP problem into two sub-problems, with these mid-points serve as the goals of the first sub-problem and the start locations of the second sub-problem. If there are mid-points that overlap, randomization is used to find alternative locations. Last, each sub-problem is solved individually, after which we stitch the solutions together. Note that, because the division is over time, there are no interactions between two sub-problems. In a k-way split, the original ILP model is effectively divided into k equal sized pieces. Solving these k pieces is usually much less time consuming than solving the single, larger ILP model. The heuristic, however, does not preserve true optimality on makespan but rather yields near-optimal solutions.

Reachability analysis Another useful, optimality preserving heuristic is reachability analysis. The basic idea here is to truncate edges from the space-time graph that are unreachable, based on the start and goal locations of each individual robot.

5 Computational Evaluation

We implemented the roadmap building phase in C++ using CGAL [4]. The discrete path planning module, written in Java, uses Gurobi [9] as the ILP solver. The experiments were carried out on an Intel i7-4850HQ laptop PC.

For evaluation, we tested of our algorithmic framework over five distinct environments. The first one is a simple square with a side length of 35 (recall that the robots are unit discs), with no internal obstacles. The rest of the environments have the same bounding square but contain different obstacle setups. We randomly select start and goal locations for all our tests. These environments, along with a typical 50-robot problem instance, are illustrated in Fig. 8.

5.1 Performance in Bounded, Obstacle-Free Environment

We first characterize how our framework performs in terms computation speed and solution optimality, as k-way split heuristic is used with different values of k. For this task, we carry out two sets of computations. The first set, covered in this sub-

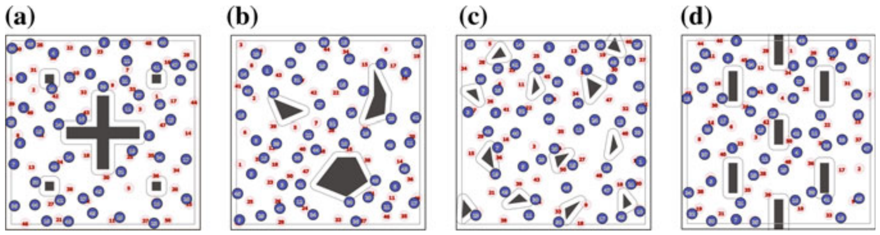


Fig. 8 Environments with obstacles and 50 start and goal locations. The labeled *blue discs* mark the start locations and the labeled *pink discs* mark the goal locations. Zoom-in on the digital version of the paper for details. **a** Plus. **b** (Halloween) Jack. **c** Triangles. **d** Bars

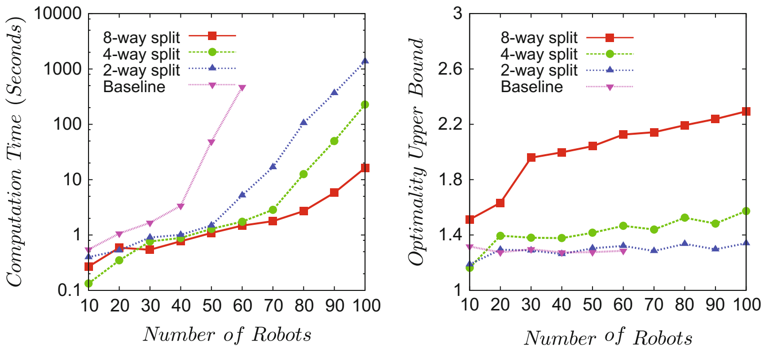


Fig. 9 Performance of framework with various choices of heuristics for a square environment without internal obstacles. *Left* Computation time. *Right* optimality ratio

section, focuses on bounded, obstacle-free environment. For this environment, we let the number of robots vary between 10–100 and evaluate the performance of the framework with the baseline algorithm (i.e., a single sub-problem), 2-way split (i.e., two sub-problems), 4-way split, and 8-way split. For each choice of the number of robots and the heuristic, 10 test cases are randomly generated sequentially and solved. The average running time and optimality ratio is plotted in Fig. 9. Note that our computation of the optimality ratio is conservative. To compute this ratio, we find the shortest distance between each pair of start and goal locations and use the maximum of these distances as the estimate of optimal time (since the robot has maximum speed of 1). We then obtain the optimality ratio by dividing the actual task completion time by the estimated value.

From the experiments, we observe that the baseline algorithm actually performs quite well for up to 40 robots in the absence of obstacles. With that said, both 2-way and 4-way splits do much better without losing much optimality–all three achieves optimality ratio between 1.2–1.6 in our experiments. With the 8-way split, sacrificing some optimality, we were able to consistently solve problems with 100 robots in 10 s on average. Such settings correspond to robots occupying over 25% of the free space, a setting that has never been attempted before in optimal multi-robot

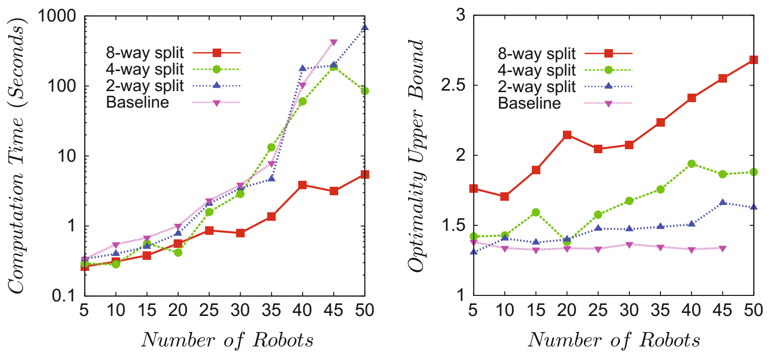


Fig. 10 Performance of our algorithmic framework with various choices of heuristics for the "Jack" environment. *Left* computation time. *Right* optimality ratio

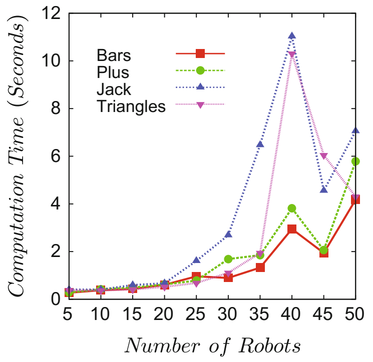
path planning. With 8-way split, problems with 125 robots in the same environment, which corresponds to a robot density over 31.4%, can be comfortably solved in about 15 min. We note that, if robot density is around 20%, our method can readily solve problems with over 300 robots (in a larger environment).

5.2 Performance in Bounded Environment with Obstacles

The second set of experiments shifts the focus to an environment with obstacles. For this we use the "Jack" environment. We choose this environment because it is in fact a relatively difficult setting as many shortest paths have to pass through the middle, causing conflicts. The experimental result, for 5–50 robots, is plotted in Fig. 10, which is consistent with our first set of experiments. We note that obstacles, while affecting the computation time, do not heavily impact the optimality of the result.

5.3 Evaluation of Overall Framework Performance

Our last set of experiments is aimed at showing the overall effectiveness of our framework. For this purpose we select the splitting heuristic automatically. Roughly, we do this by increasing k (in a k-way split) to keep each time expansion with 10 time steps, which we have found to strike a good balance between speed and optimality. For the set of environments illustrated in Fig. 8, the experimental result is plotted in Fig. 11. Our method is able to consistently solve all instances with an average solution time from 0.5 to 10 s while providing good optimality assurance on minimum makespan. The two spikes in Fig. 11a at 40 robots are due to the switching to 8-way split at 45 robot for these two environments.



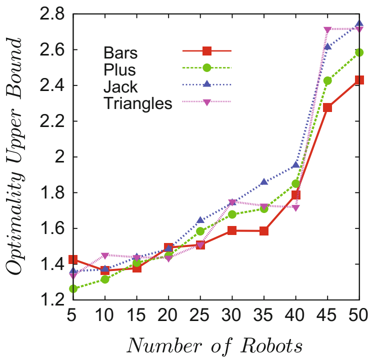


Fig. 11 Performance of the overall framework in a wide variety of environments. *Left* computation time. *Right* optimality ratio

6 Conclusion

In this paper, we present an algorithmic framework for tackling the multi-robot path planning problem in continuous, multiply-connected environments. Our framework partitions the planning task into two phases. In the first phase, the configuration space is tiled with a carefully selected regular lattice pattern, taking into account robot-robot collision avoidance. The imposed lattice is then processed to yield a roadmap that preserves the connectivity of the continuous configuration space, which is essential for achieving near optimality in the final solution. Snapping the robots and their goal locations to the roadmap then transforms the initial continuous planning problem to a discrete planning problem. In the second phase, the discrete planning problem can be solved using any graph-based multi-robot path planning algorithms, after which the solution can be readily used in continuous domains. With a good optimal planner for discrete MPP, our overall algorithm can consistently solve large problem instances with tens to hundreds of robots in seconds to minutes.

As we make an important first step here toward a generic framework for near-optimal multi-robot path planning in continuous domains with obstacles, we also bring about many natural next steps. We discuss a few of these here, which we plan to fully explore in our future research.

Nonholonomic constraints. An important issue not addressed in this paper is path planning for nonholonomic robots. We briefly touch upon this issue here. Our algorithmic framework supports quite naturally nonholonomic robots that are small-time locally controllable (STLC) with reasonable minimum turning radius. Essentially, to apply our method to a nonholonomic robot, the robot only need the capability to: *(i)* move from its start location to a nearby roadmap node with a given orientation, *(ii)* trace any path on the roadmap without incurring collision, and *(iii)* move from a roadmap node to a nearby goal location (with an arbitrary orientation). A car-like robot, or any robot that is STLC, possesses the first and the third capabilities. Then, as long as the robot has a minimum turning radius of 2, it can follow any path on a

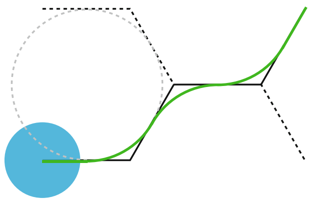


Fig. 12 A car-like robot with a mininum turning radius of 2 can trace any given path on a hexagonal lattice with side length $4/\sqrt{3}$ without violating its nonholonomic constraints or colliding with other robots

hexgonal lattice without violating its nonholonomic constraints (see Fig. 12). More importantly, multiple robots may move concurrently in such a manner without causing collisions. The introduction of nonholonomic constraints does not significantly affect optimality.

Decentralized planner. The current implementation of our framework yields a centralized algorithm. It is possible, however, to make the algorithm decentralized at the global scale. For example, we may simply let each robot perform planning individually using a method such as reciprocal velocity obstacle (RVO) based algorithm and engage locally our centralized method as the density of robots surpass some critical threshold. Note that, as the density of robots increases, RVO-based or repulsion-force-based methods generally do not have optimality guarantees and may also create deadlocks.

Optimality of hexagonal lattice in general environments. While we have shown that a hexagonal lattice structure yields the optimal tiling in the absence of obstacles, it is unclear whether this holds well when there are obstacles in the bounded environment. In future work, we plan to study this through simulation under various obstacle settings. We will also characterize the performance using lattice structures other than hexagonal ones. The reason behind this is that, although hexagonal lattice allows the highest density, each node is only 3-connected. Square lattices, for example, has a 4-connected structure, which facilitates the discrete planning phase. Generally, discrete MPP problems with higher connectivity are easier to optimally solve.

Acknowledgements This work was supported in part by ONR projects N00014-12-1-1000 and N00014-09-1-1051, and the Singapore-MIT Alliance on Research and Technology (SMART) Future of Urban Mobility project. We are grateful for the funding support that we receive. We also thank the reviewers for their insightful comments that helped significantly improve the quality of the paper.

References

1. Alonso-Mora, J.: Collaborative Motion Planning for Multi-Agent Systems. Ph.D. thesis, ETH Zurich, March 2014
2. Bien, Z., Lee, J.: A minimum-time trajectory planning method for two robots. IEEE Trans. Robot. Autom. **8**(3), 414–418 (1992)
3. Buckley, S.J.: Fast motion planning for multiple moving robots. In: ICRA (1989)
4. CGAL, Computational Geometry Algorithms Library. http://www.cgal.org

5. Erdmann, M.A., Lozano-Pérez, T.: On multiple moving objects. In: ICRA (1986)
6. Ghosh, S.K., Maheshwari, A., Pal, S.P., Saluja, S., Madhavan, C.V.: Characterizing and recognizing weak visibility polygons. Comput. Geom. **3**(4), 213–233 (1993)
7. Ghrist, R., O'Kane, J.M., LaValle, S.M.: Computing Pareto optimal coordinations on roadmaps. IJRR **24**(11), 997–1010 (2005)
8. Griffith, E.J., Akella, S.: Coordinating multiple droplets in planar array digital microfluidic systems. IJRR **24**(11), 933–949 (2005)
9. Gurobi Optimization Inc. Gurobi optimizer reference manual (2015)
10. Hopcroft, J.E., Schwartz, J.T., Sharir, M.: On the complexity of motion planning for multiple independent objects; PSPACE-hardness of the "warehouseman's problem". IJRR **3**(4), 76–88 (1984)
11. Kant, K., Zucker, S.: Towards efficient trajectory planning: the path velocity decomposition. IJRR **5**(3), 72–89 (1986)
12. Kavraki, L.E., Svestka, P., Latombe, J.-C., Overmars, M.H.: Probabilistic roadmaps for path planning in high-dimensional configuration spaces. IEEE Trans. Robot. Autom. **12**(4), 566–580 (1996)
13. Knepper, R.A., Rus, D.: Pedestrian-inspired sampling-based multi-robot collision avoidance. In: RO-MAN (2012)
14. Kornhauser, D., Miller, G., Spirakis, P.: Coordinating pebble motion on graphs, the diameter of permutation groups, and applications. In: FOCS (1984)
15. Krontiris, A., Sajid, Q., Bekris, K.: Towards using discrete multiagent pathfinding to address continuous problems. In: AAAI-WoMP (2012)
16. LaValle, S.M.: Rapidly-exploring random trees: A new tool for path planning. Technical report, Iowa State University, October 1998. Computer Science Department TR 98-11
17. LaValle, S.M., Hutchinson, S.A.: Optimal motion planning for multiple robots having independent goals. IEEE Trans. Robot. Autom. **14**(6), 912–925 (1998)
18. Lozano-Pérez, T., Wesley, M.A.: An algorithm for planning collision-free paths among polyhedral obstacles. Commun. ACM **22**(10), 560–570 (1979)
19. O'Donnell, P.A., Lozano-Pérez, T.: Deadlock-free and collision-free coordination of two robot manipulators. In: ICRA (1989)
20. Peasgood, M., Clark, C., McPhee, J.: A complete and scalable strategy for coordinating multiple robots within roadmaps. IEEE Trans. Robot. **24**(2), 283–292 (2008)
21. Peng, J., Akella, S.: Coordinating multiple robots with kinodynamic constraints along specified paths. In: Boissonat, J.-D., Burdick, J., Goldberg, K., Hutchinson, S. (eds.) Algorithmic Foundations of Robotics V, pp. 221–237. Springer, Berlin (2002)
22. Robert, W.: The Geometrical Foundation of Natural Structure. A Source Book of Design. Dover, New York (1978)
23. Ryan, M.R.K.: Exploiting subgraph structure in multi-robot path planning. J. Artif. Intell. Res. **31**, 497–542 (2008)
24. Schwartz, J., Sharir, M.: On the piano movers' problem: III. coordinating the motion of several independent bodies: the special case of circular bodies moving amidst polygonal barriers. IJRR **2**(3), 46–75 (1983)
25. Snape, J., Guy, S.J., van den Berg, J., Manocha, D.: Smooth coordination and navigation for multiple differential-drive robots. In: Khatib, O., Kumar, V., Sukhatme, G. (eds.) Experimental Robotics. Springer, Heidelberg (2014)
26. Snape, J.R.: Smooth and collision-free navigation for multiple mobile robots and video game characters. Ph.D. thesis, University of North Carolina at Chapel Hill (2012)
27. Solovey, K., Halperin, D.: k-color multi-robot motion planning. In: WAFR (2012)
28. Solovey, K., Salzman, O., Halperin, D.: Finding a needle in an exponential haystack: discrete RRT for exploration of implicit roadmaps in multi-robot motion planning. In: WAFR (2014)
29. Solovey, K., Yu, J., Zamir, O., Halperin, D.: Motion planning for unlabeled discs with optimality guarantees. In: RSS (2015)
30. Standley, T., Korf, R.: Complete algorithms for cooperative pathfinding problems. In: IJCAI (2011)

31. Švestka, P., Overmars, M.H.: Coordinated path planning for multiple robots. Robot. Auton. Syst. **23**(3), 125–152 (1998)
32. Turpin, M., Michael, N., Kumar, V.: Concurrent assignment and planning of trajectories for large teams of interchangeable robots. In: ICRA (2013)
33. van den Berg, J., Lin, M.C., Manocha, D.: Reciprocal velocity obstacles for real-time multi-agent navigation. In: ICRA (2008)
34. van den Berg, J., Overmars, M.: Prioritized motion planning for multiple robots. In: IROS (2005)
35. van den Berg, J., Snoeyink, J., Lin, M., Manocha, D.: Centralized path planning for multiple robots: optimal decoupling into sequential plans. In: RSS (2009)
36. van den Berg, J., Snape, J., Guy, S.J., Manocha, D.: Reciprocal collision avoidance with acceleration-velocity obstacles. In: ICRA (2011)
37. van der Stappen, A.F., Karamouzas, I., Geraerts, R.: Space-time group motion planning. In: WAFR (2012)
38. Wagner, G., Choset, H.: M*: a complete multirobot path planning algorithm with performance bounds. In: IROS (2011)
39. Wilson, R.M.: Graph puzzles, homotopy, and the alternating group. J. Combinat. Theo. (B) **16**, 86–96 (1974)
40. Wurman, P.R., D'Andrea, R., Mountz, M.: Coordinating hundreds of cooperative, autonomous vehicles in warehouses. AI Mag. **29**(1), 9–19 (2008)
41. Yu, J., LaValle, S.M.: Fast, near-optimal computation for multi-robot path planning on graphs. In: AAAI (2013). Late breaking papers
42. Yu, J., LaValle, S.M.: Planning optimal paths for multiple robots on graphs. In: ICRA (2013)
43. Yu, J., LaValle, S.M.: Structure and intractability of optimal multi-robot path planning on graphs. In: AAAI (2013)
44. Yu, J., Rus, D.: Pebble motion on graphs with rotations: efficient feasibility tests and planning. In: WAFR (2014)

Detecting, Localizing, and Tracking an Unknown Number of Moving Targets Using a Team of Mobile Robots

Philip Dames, Pratap Tokekar and Vijay Kumar

1 Introduction

Target detection, localization, and tracking has many applications, ranging from search-and-rescue [10] to building smart cities [21]. Consequently, such problems have long been a subject of study in the robotics community. Active target tracking typically refers to two types of tasks: estimating the trajectories of the targets from the sensor data, and actively controlling the motion of the robotic sensors to gather the data. Both problems have been studied in the literature under many different settings. Solutions have been presented for radio-based sensors [13], range-only sensors [35], bearing sensors [22], and range and/or bearing sensors [36], under centralized and decentralized settings.

Frew and Rock [9] design optimal trajectories for a single robot to track a single moving target using monocular vision. The problem of keeping targets in a robot's field-of-view can be formulated as a visual servoing problem. Gans et al. [11] design a controller which guarantees stability while keeping three or fewer targets in the field-of-view of a single mobile robot. Tracking multiple targets with multiple robots requires explicit or implicit assignment of targets to robots. Spletzer and Taylor [29] present a general solution for the multi-robot, multi-target case using a particle filter

P. Dames (✉)
Department of Mechanical Engineering,
Temple University, Philadelphia, PA 19122, USA
e-mail: pdames@seas.upenn.edu

P. Tokekar
Department of Mechanical Engineering and Applied Mechanics,
University of Pennsylvania, Philadelphia, PA 19104, USA
e-mail: tokekar@vt.edu

V. Kumar
Department of Electrical and Computer Engineering, Virginia Tech,
Blacksburg, VA 24060, USA
e-mail: kumar@seas.upenn.edu

A. Bicchi and W. Burgard (eds.), *Robotics Research*, Springer Proceedings
in Advanced Robotics 2, DOI 10.1007/978-3-319-51532-8_31

formulation. Xu et al. [33] present a mixed nonlinear integer programming formulation for assigning robots to targets as well as for determining optimal robot positioning. Recently, there has been some work on actively detecting and/or localizing an unknown number of stationary targets using radio sensors [15, 28], range-only sensors [5], and arbitrary sensor models [8].

Unlike most existing work, we study the case of tracking an *unknown* and *varying* number of *indistinguishable* targets. This introduces a number of challenges. First, we cannot maintain a separate estimator for each target, since the required number of estimators is unknown. Second, we must account for the fact that targets appear and disappear from the environment. Third, we cannot maintain a history of the target positions because we cannot uniquely identify individual targets, making prediction difficult. Finally, the system must be capable of handling false positive and false negative detections and unknown data association in addition to sensor noise. Despite these challenges, we present positive results towards solving the problem.

To solve this estimation problem we turn to Random Finite Sets (RFSs). RFSs are random variables whose realizations are finite sets. Distributions over RFSs have both a distribution over the cardinality of the set (i.e., number of targets) and a distribution over the elements of the set (i.e., position of the targets). The Probability Hypothesis Density (PHD) filter [23] is the most common estimation strategy based on RFSs. The PHD filter has recently received attention in robotics for use in robot localization [2], simultaneous localization and mapping [19], localizing static targets [8, 27], and more [1]. In this paper, we show how the PHD filter can be employed for tracking moving targets (Sect. 3.3).

An important consideration for target tracking is the motion model for the targets. A number of parametric motion models have been proposed in the literature (see [20] for a detailed survey). We employ a data-driven technique to extract the motion model, instead of assuming any parametric form. Specifically, we use Gaussian Process (GP) regression to learn a map of velocity vectors for the targets, as Joseph et al. do in their work [14]. Additionally, we show how to model the appearance and disappearance of targets within the environment (Sect. 3.2).

Next, we present a control policy to assign trajectories for all robots in order to maximize the objective function over a receding horizon. We study two objective functions using the PHD filter: mutual information and the expected number of detections by the robots. We show that both objective functions are submodular, and use a result based on [31] to prove that our greedy control policy is a 2-approximation (Sect. 3.4).

In addition to the theoretical analysis we offer, we evaluate our algorithm using simulated experiments. While our framework may be applied to a number of robot and sensor models, for the purposes of testing we restrict our attention to fixed winged aerial robots with downward facing cameras. We use a real-world taxi motion dataset from [24] for the targets and to verify our models. The simulation results reveal that robot teams using the information-based control objective track a smaller number of targets with higher precision compared to teams that maximize the expected number of detections (Sect. 4).

2 Problem Formulation

We address the problem of a team of R robots monitoring an area in order to detect, localize, and track an unknown number of moving targets using an inexpensive camera. The robots are able to localize themselves within the environment (e.g., using GPS) and robot r has pose q_t^r at time t.

The number of targets, n_t, is unknown and varies over time, since individual targets may enter and leave the area of interest. We use Random Finite Sets (RFSs) to represent the number and state of targets at any time. In the target tracking scenario, RFSs may represent either targets states or measurements. Let $X_t = \{x_{1,t}, x_{2,t}, \dots, x_{n_t,t}\}$ denote a realization of a RFS of target states at time t. A probability distribution of a RFS is characterized by a discrete distribution over the cardinality of the set and a family of densities for the elements of the set conditioned on the size, i.e.,

$$p(X = \{x_1, \dots, x_n\}) = p(|X| = n)\, p(\{x_1, \dots, x_n\} \mid |X| = n). \tag{1}$$

The first statistical moment of a distribution over a RFS is called the Probability Hypothesis Density (PHD). The PHD, $v(x)$, is a density function over the state space of an individual target with the property that the integral over any region S is the expected number of targets in that region, i.e.,

$$\int_S v(x)\, dx = E[|X \cap S|]. \tag{2}$$

The PHD filter makes the assumption that targets are independent and identically distributed and that the cardinality of the target set is characterized by a Poisson distribution. The likelihood of such an RFS is

$$p(X) = \exp\left(-\int v(x)\, dx\right) \prod_{x \in X} v(x), \tag{3}$$

which is fully characterized by the PHD.

Each robot receives a set of measurements $Z_t^r = \{z_{1,t}^r, z_{2,t}^r, \dots, z_{m,t}^r\}$ to targets that it detects within the field of view (FoV) of its sensor. The number of measurements, m_t, varies over time due to false negative and false positive detections and the motion of the robots and the targets. Let $p_d(x \mid q)$ denote the probability of a robot at q detecting a target with state x. $p_d(x \mid q) = 0$ for targets outside of the FoV of the sensor and having $p_d(x \mid q) < 1$ indicates the possibility of a false negative, or missed, detection. When a target is successfully detected, the sensor returns a measurement $z \sim g(\cdot \mid x, q)$. The sensor can also return measurements to clutter objects, causing false positive detections. Let $c(z \mid q)$ denote the PHD of clutter measurements.

The PHD filter is somewhat analogous to the Kalman filter, recursively updating the statistical moments necessary to fully characterize a distribution over the target states. Like the Kalman filter, there are two equations: the prediction and the update,

$$\bar{v}_t(x) = b(x) + \int p_s(\xi) f(x \mid \xi) v(\xi)\, d\xi \tag{4}$$

$$v_t(x) = \big(1 - p_d(x \mid q)\big)\bar{v}_t(x) + \sum_{z \in Z_t} \frac{p_d(x \mid q) g(z \mid x, q) \bar{v}_t(x)}{c(z, q) + \int p_d(\xi \mid q) g(z \mid \xi, q) \bar{v}_t(\xi)\, d\xi}. \tag{5}$$

Here $\bar{v}_t(\cdot)$ is the predicted target PHD; $b(\cdot)$ is the PHD of target births, which accounts for new targets entering the area; $p_s(\cdot)$ is the target survival probability, which accounts for targets leaving the area; and $f(\cdot \mid \xi)$ is the target motion model. In the following section, we show how to learn these parameters from a real-world dataset.

Note that the representation in the PHD filter is inherently different from more traditional target trackers. With the PHD, there is no notion of target labels or of individual target tracks. Instead, the PHD filter tracks the density of targets over time, yielding information about the bulk motion rather than about the motion of individual targets. Future work will examine the recent Labeled MeMBer filter [26], which is also based on RFSs but uses a different representation such that it is able to output labeled target tracks.

3 Target Tracking Framework

The representative problem that we consider is of a team of fixed-wing aerial robots equipped with downward-facing cameras tracking vehicles driving on the ground. However the same methodology could be extended to work with robots with other mobility constraints (e.g., ground vehicles or quadrotor platforms) and other sensor modalities (e.g., lidars or 3D depth cameras).

3.1 Sensor Parameterization

The problem of detecting vehicles using aerial imagery has been well studied [12, 34]. We use such studies to inform our selection of the sensor detection, measurement, and clutter models. The approaches presented in [12, 34] are similar, searching for image features over a range of scales in order to detect cars of different sizes or to detect cars from different elevations or with different image resolutions. In general, the system is able to have a higher detection rate if we accept a larger number of false positive detections [34, Fig. 12], [12, Fig. 8]. The detection rate may also vary with the number of pixels per target, which may be computed, using the robot pose, the approximate length scale of a target, and the image resolution, to be

$$\#\,\text{pixels per car} = \text{pixels per radian} \times \arctan \frac{\text{length of target}}{\text{distance from camera to target}}. \tag{6}$$

We assume a logistic relationship between the number of pixels per target, $n_{\mathrm{px}}(x, q)$, and the detection rate,

$$p_d(x \mid q) = p_0 + \frac{p_{d,\max} - p_{d,0}}{1 + \exp(-k(n_{\mathrm{px}}(x, q) - n_{p,0}))}, \tag{7}$$

where $p_{d,0}$, $p_{d,\max}$, k, and $n_{px,0}$ are design parameters.

The camera returns pixel (i.e., bearing) measurements to the cars detected within the image. Using the pose of the robot, we can project measurements onto the ground plane to localize the targets. The measurement model is

$$g(z \mid x, q) = \mathcal{N}(z; [r_x, c_x]^T, \sigma^2 I), \tag{8}$$

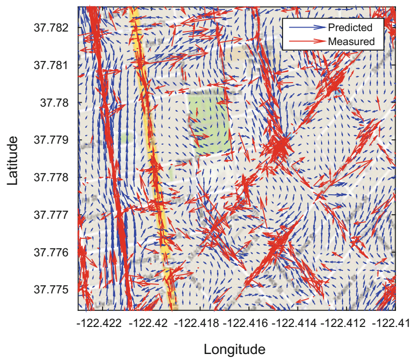
where r_x, c_x are the pixel row and column values in an image taken at q, of a target at x, σ is the standard deviation in pixels, and I is a 2×2 identity matrix.

Like the targets, the clutter is modeled as a Poisson RFS, which is completely characterized by the PHD. Without a priori knowledge of locations that are likely to have clutter, the best choice is to use a uniform distribution over the measurement space. For most computer vision-based detection algorithms, the expected number of clutter detections depends upon the detection model, with a high detection likelihood resulting in a higher detection rate [12, 34].

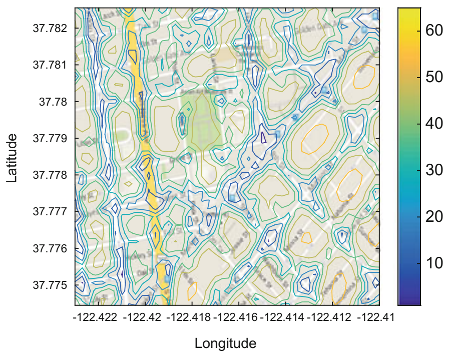
3.2 Target Parameterization

In order to predict how the target set evolves, we need models for the motion of individual targets as well as the birth/death processes of the targets. A number of motion models have been proposed in the literature, ranging from adversarial [6] to stochastic [20]. We take a data-driven approach to modeling the targets' motion, utilizing real-world datasets that are available [16]. In particular, we use Gaussian Process (GP) regression [25] to learn the function that maps the position coordinates of the targets to velocity vectors, as shown by Joseph et al. [14]. Unlike [14], we use a single GP rather than a mixture of GPs.

GP regression is a Bayesian approximation technique to learn some function $f(X)$ given measurements $y = f(x) + \varepsilon$ corrupted by Gaussian noise, $\varepsilon \sim \mathcal{N}(0, \sigma^2)$. Here, $x = [x_1, x_2]^T$ refers to the position coordinates of the targets. We learn two separate functions, f_1 and f_2, one for each axis of the ground plane, assuming that the velocities along the two axes are independent. Instead of assuming a parametric model for f_i, GP regression assumes that the joint distribution of $f_i(X)$ defined over any collection of positions, $X = \{x^1, \dots, x^k\}$, is always Gaussian. Thus, $f_i(X)$



(a) Gaussian process mean.



(b) Gaussian process standard deviation.

Fig. 1 The **a** mean and **b** standard deviation of the Gaussian Process regression motion model overlaid on the map. We only show a small patch of the environment to show the detail. The measured velocity vectors are shown in *red*, and the velocity vectors predicted over a grid are given in *blue*. The units of the velocity are m/s

is completely specified by its mean function, $m_i(X) = E[f_i(X)]$ and covariance function, $k_i(X, X') = E[(f_i(X) - m_i(X))(f_i(X') - m_i(X'))]$.

Given observed velocity vectors Y_1 and Y_2 taken at some subset of positions, X, GP regression predicts the velocity vectors at some other set of positions, X^*, as a Gaussian distribution with conditional mean and variance values [25]:

$$m_i(X^*|X) = m_i(X^*) + K_i(X^*, X)[K_i(X, X) + \sigma^2 I]^{-1}(Y_i - m_i(X))$$
$$\sigma^2_{i,X^*|X} = K_i(X^*, X^*) - K_i(X^*, X)[K_i(X, X) + \sigma^2 I]^{-1} K_i(X, X^*),$$

where $K_i(X, X')$ is a matrix whose $(m, n)^{th}$ entry is given by the covariance between $x^m \in X$ and $x^n \in X'$. We take the prior function, $m_i(X)$, to be a zero-mean distribution. Thus, if the covariance function is known, the above equations can fully predict the velocity values at arbitrary positions.

We assume that the covariance function belongs to the Matérn class with parameter $\nu = 3/2$ [25] since this choice of covariance function yields a better fit as compared to the standard squared-exponential function used by Joseph et al. [14]. The length hyperparameter of the Matérn covariance is learned using training data from the Cabspotting taxi dataset from [24]. Figure 1 shows the predicted mean and variance values given by the GP regression using the learned hyperparameter values.

We use an empirical approach to learn the target survival and birth processes. We overlay a uniform grid (1 m resolution) over the environment. Whenever a target appears in a cell, we add one to the survival count if the target was previously in another cell, we add one to the birth count if the target was previously outside the environment, and add one to the death count if at the next time step the target leaves the environment. The birth count for each cell is initialized to 10, so that the

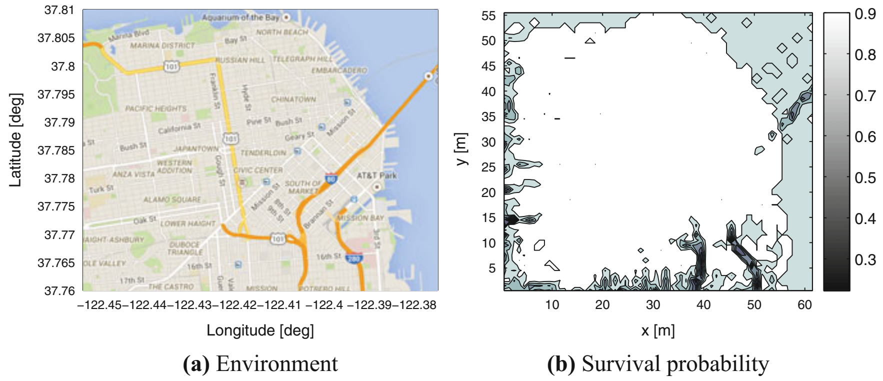


Fig. 2 **a** The area of interest, a roughly 6.15 × 5.56 km region surrounding downtown San Francisco. **b** The probability of target survival as a function of position

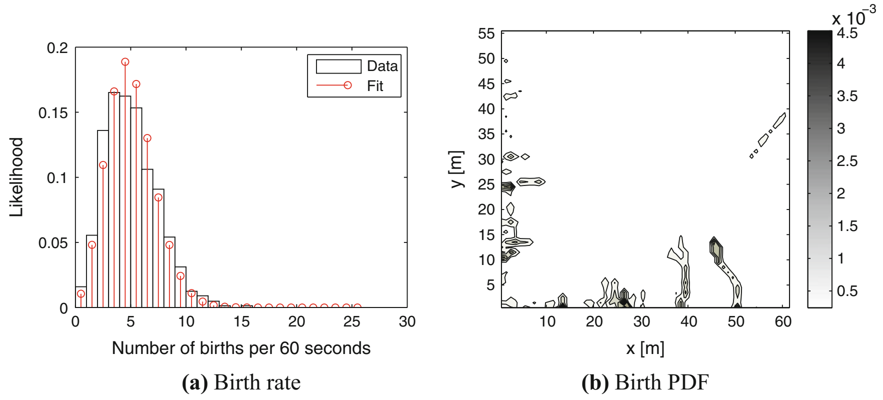


Fig. 3 Empirical target birth PHD

distribution of birth locations is uniform if there is no data. Similarly, the survival and death count for each cell are initialized to 9 and 1, respectively. The survival probability in a cell is given by the ratio of the survival count to the total survival and death counts in that cell. In the absence of data, this yields a uniform probability of survival of 0.9.

Figure 2a shows the environments used in the simulations, with the target survival probability in Fig. 2b and birth PHD in Fig. 3. As Fig. 2b shows, the targets survive with high probability in the majority of the environment. The probability decreases near the western and southern edges of the environment, where there are roads along the edge of the environment. These same areas also have the highest rates of target births, as Fig. 3b shows. One may also clearly see the highways in the southeast and the bridge in the northeast, which have the highest rates of traffic, and thus of target births and deaths. The target birth rate per minute, when considering all 536 taxis in the dataset, is 4.548 targets per minute of real time. The actual and fit birth rates are shown in Fig. 3a, with the Poisson approximation fitting the data well.

3.3 PHD Filter

We utilize the Sequential Monte Carlo (SMC) PHD filter from Vo et al. [32]. This approximates the PHD using a set of weighted particles, $v(x) \approx \sum_{i=1}^{P_t} w_i \, \delta(\mathbf{x} - \mathbf{x}_i)$. The SMC PHD filter allows for arbitrary, non-linear sensor and motion models, including a finite field of view for the sensor. New particles are added to the PHD using the birth PHD described above as well as using the most recent measurement set. A fixed number of particles, P_b, are drawn from the birth PHD and an additional P_m particles are drawn from the inverse measurement model for each measurements in the most recent set, Z_t. The weight of each of these particles is $w = \frac{\int c(z)\, dz}{P_b + |Z_t| P_m}$, where $|Z_t|$ is the cardinality of the measurement set.

3.4 Control Policy

In this section, we present our control policy for assigning trajectories for the robots. We study two objective functions for the control policy.

3.4.1 Mutual Information (MI) Objective

Mutual information is a way of quantifying the dependence between two random variables [7], and can be defined in multiple ways

$$I[\mathscr{X}, \mathscr{Z}] = \int p(X, Z) \log \frac{p(X, Z)}{p(X)p(Z)} \, dX dZ = \int \mathrm{KL}\big[p(X \mid Z) || p(X)\big] \, p(Z) \, dZ. \quad (9)$$

The last term above states that mutual information can be interpreted as the expected Kullback–Leibler divergence between the prior and the posterior, given the unknown future measurements. Thus, maximizing mutual information between the target set and the future measurements of the robots will cause the robots to take measurements that will change their belief quickly.

The robots utilize a receding horizon control policy. Each robot generates a set of candidate trajectories, with T measurements along each trajectory at evenly spaced intervals. The optimal strategy is then to choose robot trajectories that maximize the mutual information between the target set and its future measurements,

$$Q_\tau^* = \operatorname{argmax}_{Q_\tau \in Q_\tau^{1:R}} I[\mathscr{X}_{t+T}; \mathscr{Y}_\tau^{1:R} \mid Q_\tau], \quad (10)$$

where $\tau = \{t+1, \dots, t+T\}$ is the time horizon, $\mathscr{X}_{t+T}$ is the predicted location of the targets at time $t+T$, $\mathscr{Y}_\tau^{1:R}$ is the collection of binary measurements for robots 1 to R from time steps $t+1$ to $t+T$, and Q_τ are the future poses of the robots. These measurements depend on the future locations of the robots $Q_\tau = \{q_{t+1}^1, \dots, q_{t+T}^1, \dots, q_{t+T}^R\}$. Computing Q_τ^* is computationally challenging, nevertheless, we show that a greedy strategy approximates Q_τ^* by a factor of 2.

We utilize binary measurements, rather than the full measurements sets, in order to decrease the computational complexity of the control policy. This allows us to derive a closed-form expression for (10), and we have previously shown that this approach effectively drives a team of robots to detect and localize static targets [8]. The binary measurements are defined to be $y = \mathbf{1}(Z \neq \emptyset)$, where $\mathbf{1}(\cdot)$ is the indicator function. Here $y = 0$ is the event that the robot receives no measurements to any (true or clutter) objects while $y = 1$ is the complement of this, i.e., the robot receives at least one measurement. Kreucher et al. [18] take a similar approach, using a binary sensor model and an information-based objective function to schedule sensors to track an unknown number of targets.

Theorem 1 *The mutual information between the target set and the binary measurement model is a lower bound on the mutual information between the target set and the full measurement set, i.e.,* $I[\mathscr{X}; \mathscr{Y} \mid Q] \leq I[\mathscr{X}; \mathscr{Z} \mid Q]$.

Proof Note that y is deterministically related to Z, $y = \mathbf{1}(Z \neq \emptyset)$. This allows us to apply the Data Processing Inequality [7, Theorem 2.8.1], which states that functions of the data cannot increase the amount of information. □

We utilize a greedy approximation strategy to evaluate (10), similar to that used by Tokekar et al. [31]. Using this approach, each robot computes the utility of each action according to (10). The robot and action with the highest utility are selected. The remainder of the team then plans again, conditioned on the action of the first robot, and the robot and action with the highest utility are again selected. This process repeats until all robots have been assigned an action. Using the fact that mutual information is a submodular set function of robot poses, we can show that this greedy assignment policy is a 2-approximation.

Lemma 1 $I[\mathscr{X}; \mathscr{Y} \mid Q]$ *is a submodular set function of* Q.

Proof See [17, Proposition 2]. □

Theorem 2 *Let* Q^G *be the robot poses selected by the greedy assignment policy and* Q^* *be the robot actions selected by the full, joint evaluation of* (10). *Then greedy is a 2-approximation, i.e.,* $I[\mathscr{X}; \mathscr{Y} \mid Q^G] \geq \frac{1}{2} I[\mathscr{X}; \mathscr{Y} \mid Q^*]$.

Proof It is known that the greedy algorithm yields a 2-approximation for maximizing a monotone, submodular function subject to a partition matroid constraint [4]. We can create a set system using the candidate robot actions, as shown in [31]. This set system defines a partition matroid, which along with the previous lemma proves the desired result. □

Atanasov et al. [3] recently proved the same bound holds for the centralized and the decentralized case.

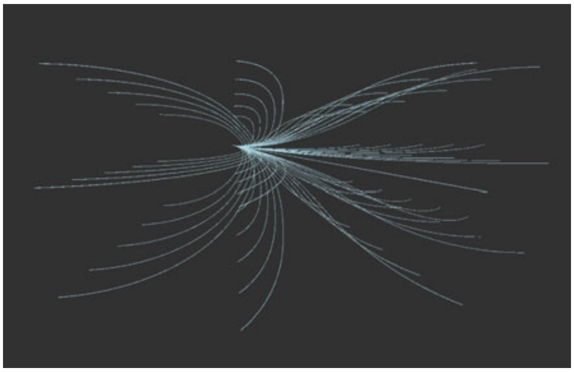


Fig. 4 Sample UAV trajectories

3.4.2 Expected Number of Detections (END) Objective

The Expected Number of Detections (END) objective function is given by

$$N[X \mid Q] = \int \left(1 - \prod_{q \in Q} \big(1 - p_d(x \mid q)\big)\right) v(x)\, dx. \tag{11}$$

This objective gives the expected number of targets detected by at least one robot, and is a submodular set function of Q so the greedy assignment algorithm will be a 2-approximation, similar to the previous theorem.

Lemma 2 *The END objective function, $N[X \mid Q]$, is a submodular function of Q.*

Proof The difference in the objective when adding a single robot is

$$N[X;\, Q \cup \{q'\}] - N[X;\, Q] = \int p_d(x \mid q') \prod_{q \in Q} (1 - p_d(x \mid q)) v(x)\, dx.$$

For any $R \subseteq Q$, the product $\prod_{r \in R}(1 - p_d(x \mid r)) \geq \prod_{q \in Q}(1 - p_d(x \mid q))$ since $p_d(x \mid q) \in [0, 1], \forall x, q$. Thus $N[X;\, R \cup \{q'\}] - N[X;\, R] \geq N[X;\, Q \cup \{q'\}] - N$ $[X;\, Q]$, so by definition $N[X, Q]$ is submodular. □

3.4.3 Trajectory Generation

We use a simple model for a fixed-wing aircraft with three basic control inputs: forward velocity, yaw rate, and pitch rate. For each control input we select a range of possible values. For each possible set of control inputs we integrate the position, yaw, and pitch forward in time using a 1-step Euler integration scheme. Any trajectories that bring the robots above or below the elevation limits are discarded as invalid, as are any that result in collision. The remaining trajectories are interpolated to yield the T poses at which each robot will take a measurement. Figure 4 shows sample trajectories for a single robot.

4 Results

To test the performance of the different objective functions, we ran a series of experiments using simulated robots, varying the number of robots, the length of the planning horizon, the objective function, and the target motion model. We used teams of 2, 4, and 6 robots, keeping the number of planning steps constant ($T = 2$) and keeping the total number of actions for the team constant ($RT = 12$). We perform 5 trials with each configuration, randomly selecting a subset of 80 targets from the taxi database for the ground truth target motion in each trial. Note that the true number of targets in the area of interest varies over time as targets enter and leave. The robots monitor the area from Fig. 2a, which is scaled down by a factor of 100 from the real world. We also sped up the data by a factor of 60, so 1 s in simulation represents 1 min of real time, in order to speed up the simulations. The data is taken from 4–9 pm on May 18, 2008, a time of day where there will be plenty of taxi traffic.

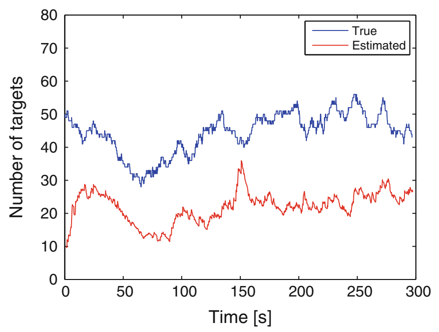
It is worth noting that a number of competing multi-target tracking methods exist [30], most notably the Multiple Hypothesis Tracker (MHT) and Joint Probabilistic Data Association (JPDA). However, to the best of our knowledge, there do not exist active multi-robot control policies based on these estimation algorithms. This makes comparisons to these methods beyond the scope of this paper.

4.1 *Moving Targets*

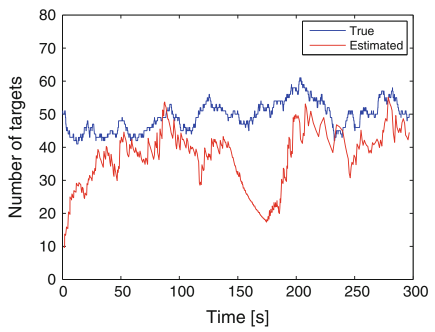
The two target motion models that we consider are the Gaussian Process (GP) described in Sect. 3.2, and a Gaussian random walk (GRW) model. In GRW we model the target as performing a random walk, with a velocity drawn at random from a Gaussian distribution. Note that these models are used only to update the PHD; the actual targets trajectories are given by the taxi dataset. In both cases we use the survival and birth processes described in Sect. 3.2, with the birth rate set to 0.6788 to account for the reduced number of data files used.

Figures 5, 6 and 7 show how the ratio of the expected number of targets to true targets, the robot elevation, and the target set entropy change during a single run. These are representative trials of a team of 2 robots with a planning horizon of 6 time steps.

Figures 8, 9 and 10 show the ratio of the expected number of targets to true number of targets, the fraction of true targets within the sensor FoV, and the ratio of expected targets to true targets within the sensor FoV, respectively. In general, the fraction of targets tracked by the team depends much more on the motion model and the

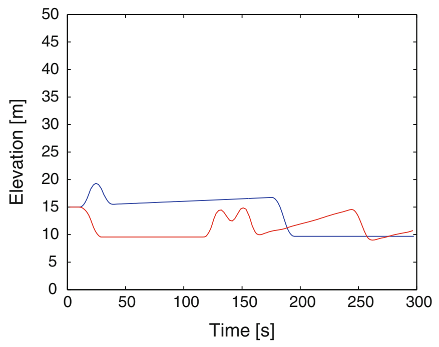


(a) GP motion model with MI objective

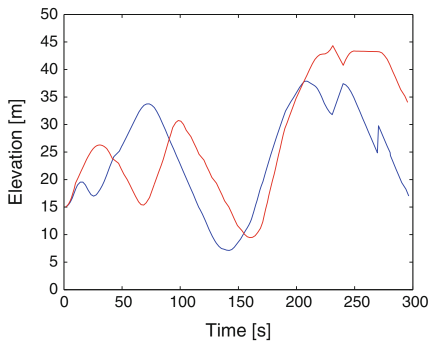


(b) GP motion model with END objective

Fig. 5 Ratio of the expected number to the true number of targets over a single run for $R = 2$ and $T = 6$

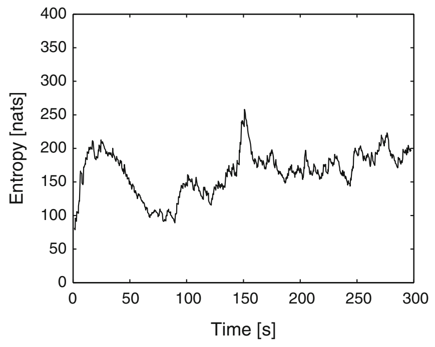


(a) GP motion model with MI objective

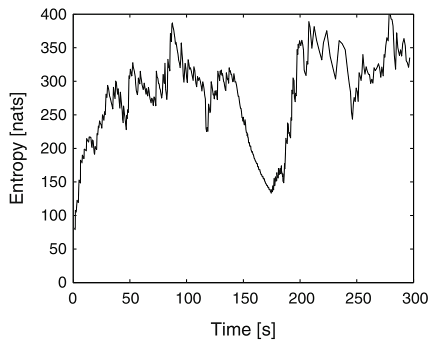


(b) GP motion model with END objective

Fig. 6 The elevation of the robots over a single run for $R = 2$ and $T = 6$

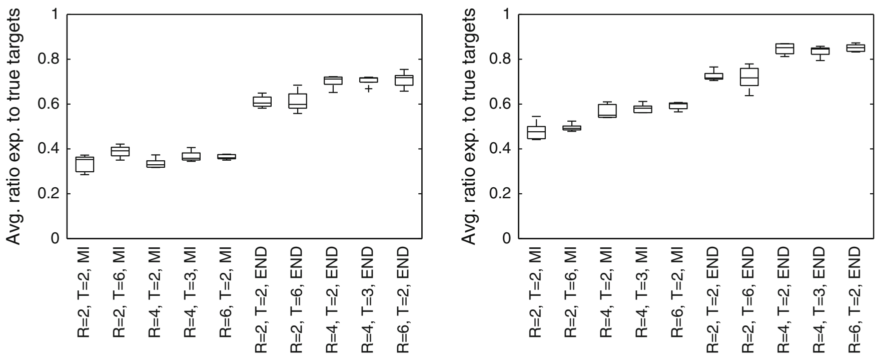


(a) GP motion model with MI objective



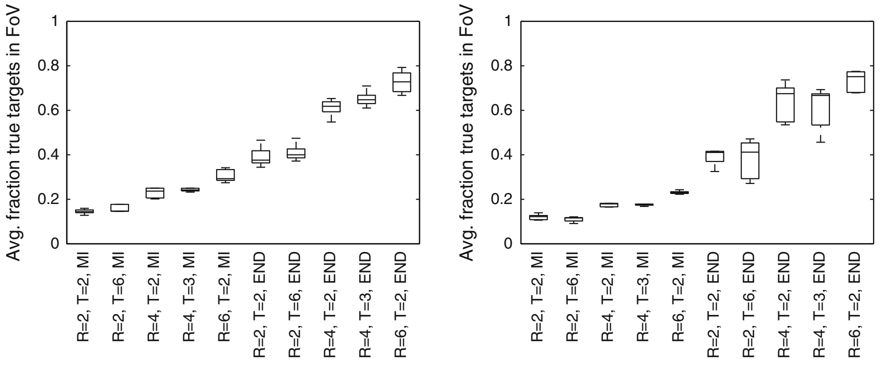
(b) GP motion model with END objective

Fig. 7 The entropy of the target set over a single run for $R = 2$ and $T = 6$



(a) Gaussian random walk **(b)** Gaussian process

Fig. 8 Average ratio of the expected number to the true number of targets over a single run



(a) Gaussian random walk **(b)** Gaussian process

Fig. 9 Average fraction of the number of targets within the team's field of view over a single run

objective function than on the team size or planning horizon, despite the fact that larger teams and planning horizons cause the robots to observe a larger number of targets. Additionally, the ratio of the expected number of targets to the true number of targets within the sensor FoV is largely independent of the objective function, team size, or planning horizon.

Overall, the robot teams using the information based control objective (MI) estimate and track fewer targets than the teams using the END objective but each target is tracked with higher quality. Additionally, the teams using the GP-based motion model track more targets than those using the GRW motion model.

The reason for this difference is due to the emergent behavior of the different control objectives. Robots using the MI objective tend to stay closer to the ground in order to decrease uncertainty in the location of individual targets. On the other hand, robots using the END objective fly at a higher altitude, as Fig. 11 shows. Note that increasing the altitude decreases the probability of detection, while increasing the

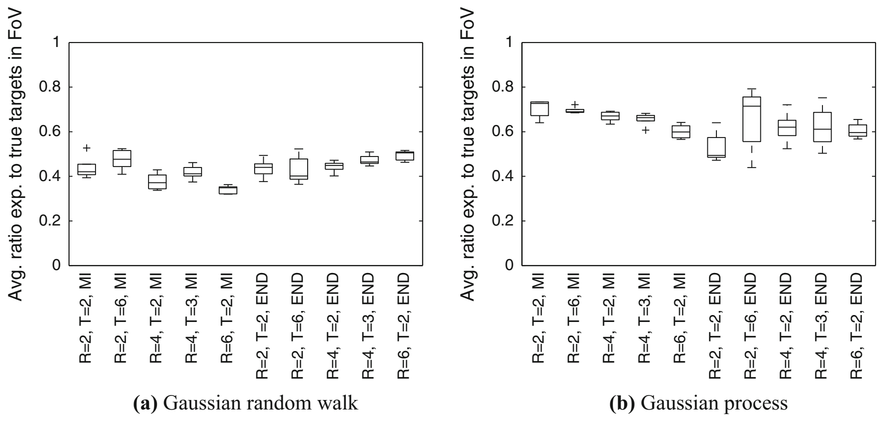


Fig. 10 Average ratio of the expected number to the true number of targets within the team's field of view over a single run

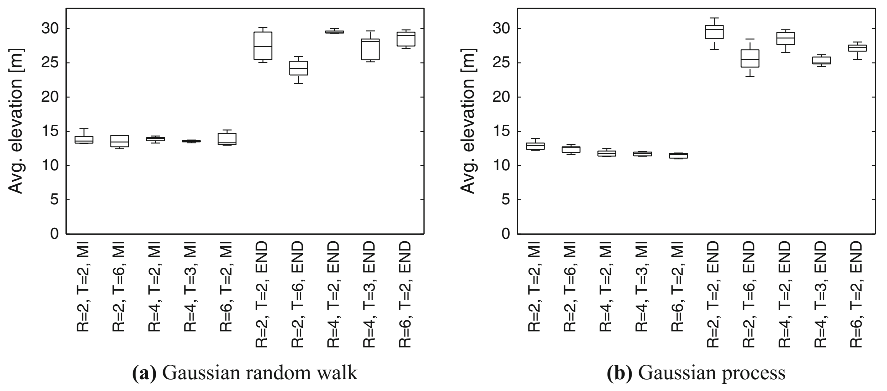
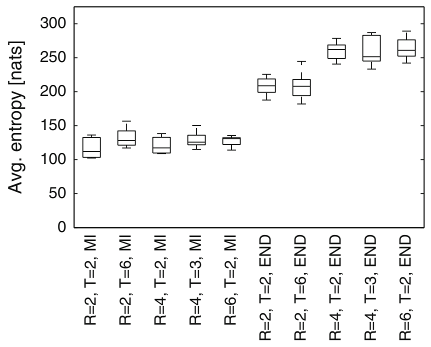


Fig. 11 Average elevation of the robots over a single run

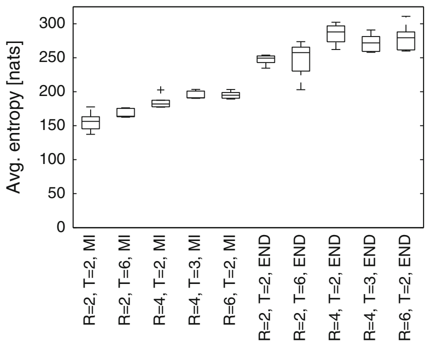
sensor FoV. Consequently, flying to the highest altitude is not necessarily optimal. Figure 12 shows the average target entropy. This is substantially lower for the teams using MI, indicating that the targets are being tracked with less uncertainty.

4.2 Static Targets

We also test the performance of our framework with static targets using a team of 4 robots with a planning horizon of 3. The simulation parameters are identical, except we replace the 80 taxi data traces with 80 randomly drawn static target locations. The resulting final estimated number of targets and target entropies are shown in Fig. 13. The final estimated number of targets is very close to 1 using both objective

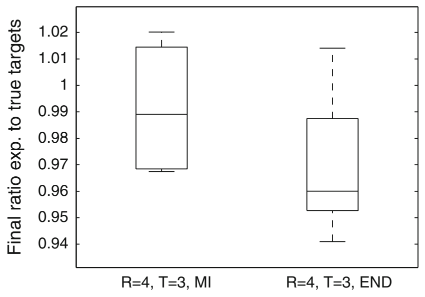


(a) Gaussian random walk

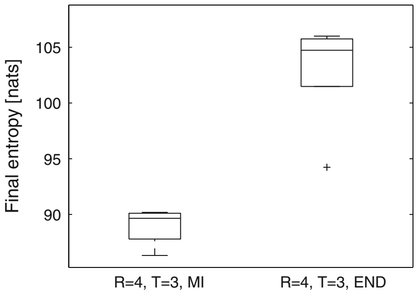


(b) Gaussian process

Fig. 12 Average entropy of the target set over a single run



(a) Final ratio of the expected number of targets to the true number of targets.



(b) Final entropy of the estimated target set.

Fig. 13 Performance of our framework with static targets

functions, indicating that the system is able to correctly determine the number of targets. The entropy is also lower than in the case of moving targets.

5 Conclusions

In this paper we describe a framework for detecting, localizing, and tracking an unknown number of moving targets using a team of mobile robots. The robot team uses the Probability Hypothesis Density filter to simultaneously estimate the number of targets and the states of the targets. The PHD filter is robust to false negative and false positive detections and sensor noise and does not require any explicit data association. Using the estimate of the target set from the PHD filter, the robots greedily select actions that maximize submodular control objectives. The two control objectives that we consider in this paper are the expected number of detected (END) targets by the team and the mutual information (MI) between the predicted targets and the future detections of the robots. We validate our framework through extensive simulations using a real-world dataset for target motion. Robot teams using the END objective track a higher fraction of the targets but do not localize the targets with high precision. Conversely, robot teams using MI track a smaller number of targets but have significantly lower uncertainty in the target positions.

Acknowledgements This work was funded by ONR MURI Grants N00014-07-1-0829, N00014-09-1-1051, and N00014-09-1-1031, ARO Grant W911NF-13-1-0350, NSF Grant IIS-1426840, and TerraSwarm, one of six centers of STARnet, a Semiconductor Research Corporation program sponsored by MARCO and DARPA. P. Dames was supported by the Department of Defense through the National Defense Science & Engineering Graduate Fellowship (NDSEG) Program.

References

1. Adams, M., Vo, B.-N., Mahler, R. (eds.): Advances in probabilistic modeling: applications of stochastic geometry. IEEE Robot. Autom. Mag. 21(2) (2014). IEEE
2. Atanasov, N., Zhu, M., Daniilidis, K., Pappas, G.J.: Semantic localization via the matrix permanent. In: Robotics Science and Systems (2014)
3. Atanasov, N., Le Ny, J., Daniilidis, K., Pappas, G.: Decentralized active information acquisition: theory and application to multi-robot slam. In: IEEE International Conference on Robotics and Automation (ICRA) (2015)
4. Calinescu, G., Chekuri, C., Pál, M., Vondrák, J.: Maximizing a submodular set function subject to a matroid constraint. In: Integer Programming and Combinatorial Optimization, pp. 182–196. Springer (2007)
5. Charrow, B., Michael, N., Kumar, V.: Active control strategies for discovering and localizing devices with range-only sensors. In: Workshop on Algorithmic Foundations in Robotics (WAFR) (2014)
6. Chung, T.H., Hollinger, G.A., Isler, V.: Search and pursuit-evasion in mobile robotics. Auton. Robot. **31**(4), 299–316 (2011)
7. Cover, T., Thomas, J.: Elements of Information Theory. Wiley, New York (2012)
8. Dames, P., Kumar, V.: Autonomous localization of an unknown number of targets without data association using teams of mobile sensors. IEEE Trans. Autom. Sci. Eng. **12**(3), 850–864 (2015)
9. Frew, E.W., Rock, S.M.: Trajectory generation for constant velocity target motion estimation using monocular vision. In: IEEE International Conference on Robotics and Automation, pp. 3479–3484 (2003)
10. Furukawa, T., Bourgault, F., Lavis, B., Durrant-Whyte, H.F.: Recursive Bayesian search-and-tracking using coordinated UAVs for lost targets. In: IEEE International Conference on Robotics and Automation, pp. 2521–2526 (2006)
11. Gans, N.R., Hu, G., Nagarajan, K., Dixon, W.E.: Keeping multiple moving targets in the field of view of a mobile camera. IEEE Trans. Robot. **27**(4), 822–828 (2011)
12. Grabner, H., Nguyen, T.T., Gruber, B., Bischof, H.: On-line boosting-based car detection from aerial images. ISPRS J. Photogramm. Remote Sens. **63**(3), 382–396 (2008)
13. Hollinger, G.A., Djugash, J., Singh, S.: Target tracking without line of sight using range from radio. Auton. Robot. **32**(1), 1–14 (2012)
14. Joseph, J., Doshi-Velez, F., Huang, A.S., Roy, N.: A bayesian nonparametric approach to modeling motion patterns. Auton. Robot. **31**(4), 383–400 (2011)
15. Kim, C.-Y., Song, D., Xu, Y., Yi, J., Wu, X.: Cooperative search of multiple unknown transient radio sources using multiple paired mobile robots. IEEE Trans. Robot. **30**(5), 1161–1173 (2014)

16. Kotz, D., Henderson, T.: CRAWDAD: a community resource for archiving wireless data at dartmouth. IEEE Pervasive Comput. **4**(4), 12–14 (2005)
17. Krause, A., Guestrin, C.E.: Near-optimal nonmyopic value of information in graphical models. In: Conference on Uncertainty in Artificial Intelligence (2005)
18. Kreucher, C., Kastella, K., Hero III, A.O.: Sensor management using an active sensing approach. Signal Process. **85**(3), 607–624 (2005)
19. Leung, K., Inostroza, F., Adams, M.: Evaluating set measurement likelihoods in random-finite-set SLAM. In: IEEE International Conference on Information Fusion, pp. 1–8. IEEE (2014)
20. Li, X.R., Jilkov, V.P.: Survey of maneuvering target tracking. part i. dynamic models. IEEE Trans. Aerosp. Electron. Syst. **39**(4), 1333–1364 (2003)
21. Li, X., Lu, R., Liang, X., Shen, X., Chen, J., Lin, X.: Smart community: an internet of things application. IEEE Commun. Mag. **49**(11), 68–75 (2011)
22. Logothetis, A., Isaksson, A., Evans, R.J.: An information theoretic approach to observer path design for bearings-only tracking. In: IEEE International Conference on Decision and Control, vol. 4, pp. 3132–3137 (1997)
23. Mahler, R.: Multitarget bayes filtering via first-order multitarget moments. IEEE Trans. Aerosp. Electron. Syst. **39**(4), 1152–1178 (2003)
24. Piorkowski, M., Sarafijanovic-Djukic, N., Grossglauser, M.: CRAWDAD data set epfl/mobility (v. 2009-02-24). Downloaded from http://crawdad.org/epfl/mobility/, February 2009
25. Rasmussen, C., Williams, C.: Gaussian processes for machine learning (2006)
26. Reuter, S., Vo, B.-T., Vo, B.-N., Dietmayer, K.: The labeled multi-bernoulli filter. IEEE Trans. Signal Process. **62**(12), 3246–3260 (2014)
27. Ristic, B., Vo, B.-N., Clark, D.: A note on the reward function for PHD filters with sensor control. IEEE Trans. Aerosp. Electron. Syst. **47**(2), 1521–1529 (2011)
28. Song, D., Kim, C.-Y., Yi, J.: Simultaneous localization of multiple unknown and transient radio sources using a mobile robot. IEEE Trans. Robot. **28**(3), 668–680 (2012)
29. Spletzer, J.R., Taylor, C.J.: Dynamic sensor planning and control for optimally tracking targets. Intl. J. Robot. Res. **22**(1), 7–20 (2003)
30. Stone, L.D., Streit, R.L., Corwin, T.L., Bell, K.L.: Bayesian Multiple Target Tracking. Artech House, Norwood (2013)
31. Tokekar, P., Isler, V., Franchi, A.: Multi-target visual tracking with aerial robots. In: IEEE/RSJ International Conference on Intelligent Robots and Systems, pp. 3067–3072. IEEE (2014)
32. Vo, B.-N., Singh, S., Doucet, A.: Sequential monte carlo methods for multi-target filtering with random finite sets. IEEE Trans. Aerosp. Electron. Syst. **41**(4), 1224–1245 (2005)
33. Xu, Z., Fitch, R., Underwood, J., Sukkarieh, S.: Decentralized coordinated tracking with mixed discrete-continuous decisions. J. Field Robot. **30**(5), 717–740 (2013)
34. Zhao, T., Nevatia, R.: Car detection in low resolution aerial images. Image Vis. Comput. **21**(8), 693–703 (2003)
35. Zhou, K., Roumeliotis, S.I.: Optimal motion strategies for range-only constrained multisensor target tracking. IEEE Trans. Robot. **24**(5), 1168–1185 (2008)
36. Zhou, K., Roumeliotis, S.I.: Multirobot active target tracking with combinations of relative observations. IEEE Trans. Robot. **27**(4), 678–695 (2011)