Auton Robot (2012) 33:89–102
DOI 10.1007/s10514-012-9289-9

Adaptive fast open-loop maneuvers for quadrocopters

Sergei Lupashin · Raffaello D'Andrea

Received: 31 July 2011 / Accepted: 15 March 2012 / Published online: 7 April 2012

Abstract We present a conceptually and computationally lightweight method for the design and iterative learning of fast maneuvers for quadrocopters. We use first-principles, reduced-order models and we do not require nor make an attempt to follow a specific state trajectory—only the initial and the final states of the vehicle are taken into account. We evaluate the adaptation scheme through experiments on quadrocopters in the ETH Flying Machine Arena that perform multi-flips and other high-performance maneuvers.

Keywords Aerial robotics · Aerobatics · Learning · Policy gradient

1 Introduction

Our goal is to create a method for performing fast adaptive maneuvers for quadrocopters from loose maneuver definitions that is straightforward to implement and to understand. An example of some of the demonstrated results of the resulting method is pictured in Fig. 1.

The growing ubiquity of small, robust, affordable and highly dynamic micro aerial vehicles (MAVs) such as quadrocopters in research labs has yielded impressive demonstrations of high-performance aerial motion control. Small autonomous helicopters, quadrocopters and other aerial vehicles have gone a long way from the hovering regime and are now able to perform various impressive high-performance maneuvers (Mellinger et al. 2010; Ritz et al. 2011) and acrobatics (Abbeel et al. 2010; Gillula et al. 2009; Lupashin and D'Andrea 2011; Gerig 2008).

Fig. 1 Composite time-lapse photo of a quadrocopter performing a triple flip designed and learned using the described algorithm. The individual snapshots are offset horizontally for clarity—the vehicle returns to the original spot in the actual maneuver. The maximum commanded turn rate is 1800°/s. The complete hover-to-hover maneuver takes 1.6 seconds

Extreme aerobatic maneuvers provide for stunning and readily accessible demonstrations of MAV capabilities and are a great motivator for improving our understanding of MAV flight and control. Existing demonstrations rely on combinations of careful tuning, empirical system identification, and adaptation/learning to enable the vehicle to perform the motion accurately and reliably. As a result many of the methods are quite complex, algorithmically and compu-

This work was supported in part by the Swiss National Science Foundation through the National Centre of Competence in Research Robotics.

S. Lupashin (✉) · R. D'Andrea
Institute for Dynamic Systems and Control, ETH Zurich,
Sonneggstr. 3, ML K 36.2, 8092 Zurich, Switzerland
e-mail: sergeil@ethz.ch

tationally. A key difficulty is the non-existence of accurate analytical models for many flight regimes of helicopters and other aerial vehicles.

There have been efforts to precisely model and characterize rotary-wing flight in high-performance regimes for quadrocopters such as by Hoffmann et al. (2011), preceded by decades of theoretical and applied research on full-scale helicopters. As a result there exist various models specialized for certain conditions such as hover, certain axial motions, autorotation, and translational movement with constant velocity. As an added complication, it has been shown that many accepted theoretical or empirical models for full-sized helicopters break down for smaller vehicles as the associated Reynolds numbers are on the order of 10^4 to 10^5, which is different by orders of magnitude from full-sized rotorcraft (Leishman 2006).

Yet high-performance maneuvers such as aerobatics are analogous to quick, fast tasks performed by living things that are critical to everyday function. For example, as humans we need to be able to quickly grasp and manipulate objects, or to reflexively react to threats and sudden surprises without waiting for high-level feedback. Flying vehicles will need similar skills to survive in real environments, making fast, dynamic flight not just impressive but useful and even necessary. By developing robust algorithms for aerobatics we are also developing methodologies for performing motions vital to the success of typical MAVs in future real-world applications.

We look to human motor control for parallels. Motor motion learning is commonly split into two components: structural and parametric learning (Wolpert and Flanagan 2010). Structural learning involves learning the general outline of a problem such as which muscle groups to use or the general form of a motion, while parametric learning is episodic and attempts to use given knowledge about the problem structure combined with trials to improve performance. For example, performing an already known tennis serve with a different racket is a parametric learning task, while learning to make a serve for the first time is a structure learning task (Wolpert et al. 2001). During parametric motion learning we intuitively focus on adjusting just a few "parameters", which we can often identify semi-explicitly. We attempt to loosely follow and exploit the structural-parametric split in motion learning by providing *a priori* explicit structural motion information, by selecting intuitive parameters to adjust, and by using an automated method for the parametric learning.

This work presents both a methodology and a specific algorithm for designing parameterized high-performance maneuvers for quadrocopters along with a scheme for iteratively improving the performance of these motions from experiments. A high-level overview of this method is pictured in Fig. 2.

The method presented in this work has been applied successfully to several maneuvers such as flips and near-time

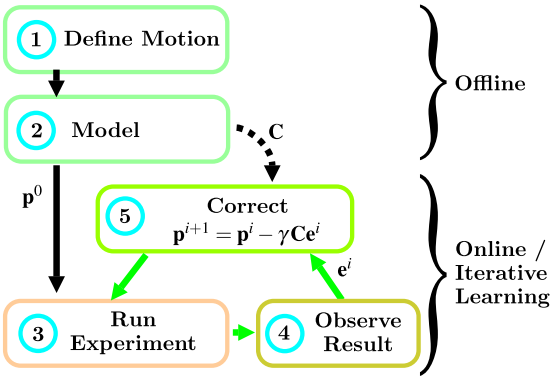

Fig. 2 Overview of the described motion design and iterative adaptation method. $\mathbf{p}$ represents the parameters to be adapted, $\mathbf{C}$ is a first-order correction matrix, γ is a correction step size, and $\mathbf{e}$ is a vector of error measurements. (*1*) The user defines a motion in terms of initial and desired final states and a parameterized input function. (*2*) A first-principles continuous-time model is used to find nominal parameters $\mathbf{p}^0$ and $\mathbf{C}$. (*3*) The motion is performed on the physical vehicle, (*4*) the error is measured and (*5*) a correction is applied to the parameters. The process is then repeated

optimal translation. Lupashin et al. (2010) first described this method in 2D for adaptive flips; Lupashin and D'Andrea (2011) expanded on it by controlling all of the degrees of freedom and introducing another maneuver. In Ritz et al. (2011) we also demonstrated how the method was applied for a completely different motion to enable near-time-optimal benchmarking on physical flying vehicles.

This work further expands on the details of the adaptation method used in these experiments and discusses experiences in using it both as a tool to enable performance of novel aerobatic maneuvers and as a way to bring theoretical results into the physical world.

We attempt to keep the presented methodology as general as possible; we believe that it can be readily applied to a variety of problems outside of quadrocopter aerobatics. The method leaves the freedom to select any variables as errors or as parameters, and input trajectories can be defined at different levels: accelerations, velocities, reference positions, and so on.

This work is organized as follows: We begin by reviewing and relating the proposed method to other published results in Sect. 2. In Sects. 3–5 we describe the overall algorithm. In Sect. 6 we introduce the quadrotor flying vehicles used for these maneuvers and the first-principles models for vehicle dynamics and the onboard controllers. We then introduce some maneuver design and parameter reduction tools specific to quadrocopters in Sect. 7. In Sect. 8 we apply the method to performing fast flips, first in a simplified 2D manner and then for learning errors in all degrees of freedom. In Sect. 9 we show how the algorithm was used to enable the transport of near-time-optimal maneuvers from simulation to physical vehicles. In Sect. 10 we briefly describe the experimental platform and show experimental results. Finally,

we address limitations and present an outlook and concluding remarks in Sect. 11.

2 Related work

High-performance rotary-wing aerobatic flight has been explored by several research groups using indoor and outdoor helicopters and quadrocopters. Since it is not tractable to predict many of the effects governing high-performance helicopter or quadrocopter flight, learning and adaptation algorithms are typically used to achieve the desired performance without excessive empirical modeling.

Abbeel et al. (2010) demonstrated how a nominal aerobatic trajectory can be extracted from several demonstrations by a human pilot. The same demonstrations are also used to construct families of locally accurate models that enable feedback control throughout the aerobatic maneuver. This is a sophisticated approach with great flexibility and wide applicability, necessarily requiring good understanding of the underlying mathematics and the various processing parameters. At the core of this approach is a critical dependency on being able to performing a maneuver manually before a trajectory and a controller is able to be synthesized to repeat the motion. In this work we take a different path of explicitly avoiding prior demonstrations, as we wish to demonstrate a more loosely-defined maneuver but at speeds that require precision timing that exceeds human capabilities.

Another family of approaches are commonly classified as Iterative Learning Control (ILC). Purwin and D'Andrea (2011) demonstrated such an approach for learning fast translation for a larger-scale quadrocopter. This is done by first calculating a nominal trajectory and a nominal discretized input trajectory. The vehicle then attempts to perform the motion and the input trajectory is adjusted by a numerical optimization that uses a nominal first-principles model. This approach has the advantage that an exact nominal trajectory can be specified and if it is physically feasible, and the initial attempt is close enough, the vehicle should end up learning to follow that exact trajectory. New motions can be learned by extension by slowly changing the reference trajectory while repeating the learning process.

The method described in this work is an almost perfectly polar opposite to ILC methods: we are not able to control the exact learned trajectory and leave it up to the physical vehicle to "find" the real state trajectory to perform a given maneuver. This provides an alternative learning scheme where we avoid having to weigh errors over the trajectory or to perform complex correction steps. ILC provides a very strong tool for precisely following a predetermined trajectory; this method can be used to actually find such "nominal" trajectories for flight regimes where modeling or simulation fails. Note that while the nominal trajectory is unspecified, the maneuver designer is still asked to provide other *a priori* knowledge including the form of the input trajectory and an initial guess for the values being learned.

An approach related to the one described in this work was taken for performing aggressive quadrocopter maneuvers by Mellinger et al. (2010). Specific maneuvers were split up into component stages and intuitive gradient-like correction laws were implemented to improve performance. The method presented here is similar in structure with the main difference that numerical methods are used to find both nominal parameters and to calculate gradients.

Other recent related work includes provably-safe backflips by Gillula et al. (2009) and auto-generated outdoor aerobatic helicopter shows by Gerig (2008). In both cases the maneuvers were shown outdoors and were not sensitive to the exact repeatability and accuracy required for maneuvers in an indoor space.

3 Method overview

The method takes the following form, split into offline preparation steps performed once and an iterative online experiment/correction step that is performed repeatedly:

Step 1: offline: define maneuver

(a) Define maneuver as a desired initial state and a desired final state. The entire state vector may not be relevant here: for example, lateral errors (y, $\dot{y}$, etc.) could be ignored if a maneuver is formulated in the (x,z) plane.
(b) Pick reference level (acceleration, angular rates, etc.) and define a parameterized control reference trajectory from intuitive analysis of maneuver structure. Consider the most basic actions that need to be performed to complete the maneuver, and the core variables that parameterized these actions.
(c) Optionally use algebraic constraints and tools such as time-optimal control to create algebraic links between parameters and to reduce the final count of free parameters.

Step 2: offline: find nominal motion and correction matrix

(a) Use a reduced-order model combined with an initial guess of the parameters $\mathbf{p}^g$ and a numeric solver to find nominal parameters $\mathbf{p}^0$ and trajectory.
(b) Calculate Jacobian about nominal parameter set. Calculate correction matrix $\mathbf{C}$ such that $\mathbf{C}$ is a right inverse of $\mathbf{J}$.

Step 3: online: adaptation through experiments

(a) Use parameterized input function to generate input trajectory from current parameter set $\mathbf{p}^i$.
(b) Run experiment and observe the error $\mathbf{e}^i$ between the actual and the desired final state.
(c) Apply correction: $\mathbf{p}^{i+1} = \mathbf{p}^i - \gamma\, \mathbf{C}\mathbf{e}^i$.
(d) If parameter constraints violated, temporarily reduce correction until new parameter set is allowable.
(e) Modify step size γ if desired. For experiments in this work, $\gamma^i = \max(1/i, 0.1)$ where i starts at 1.
(f) Repeat Step 3.

4 Defining maneuvers and finding nominal parameters

We define a maneuver as an initial state, a desired final state, and a parameterized control function to be used for the duration of the maneuver. The parameterized control function represents a family of input functions that contain one or more specific input trajectories that will, on average, drive the system from the initial state to the desired final state. An analogous task in human motor learning could be: the initial state of a basketball being in the hands of a player, the desired final state of the ball going through the hoop, and the general idea of the form of the solution: bending the elbows, accelerating the ball, and releasing it.

More formally, we desire to drive a physical system from an initial state to a desired final state x^* using an input function of the form $\mathcal{U}(t, \mathbf{p})$, parameterized in terms of some parameters $\mathbf{p}$. To simplify the discussion, assume that the given problem is possible: that is, there is one or more ideal parameter set $\mathbf{p}^*$ that will drive the system sufficiently close to x^* for the maneuver to be deemed successful; the objective is to find this parameter set without knowing all of the details of the physical system dynamics. In addition, assume the initial state of the system to be exactly the desired initial state of the maneuver, which for the examples described later in this work is fixed hover.

We start by constructing a nominal first-principles model like the one described in Sect. 6. We then come up with an initial guess for the parameters, $\mathbf{p}^g$. Later in the work we will use gross algebraic approximations to calculate this initial parameter guess.

We refer to the final state of the nominal system after executing a parameterized maneuver starting from the nominal initial state as $\Phi(\mathbf{p}, \hat{\mathbf{s}})$, where $\mathbf{p}$ is the parameter set used, and $\hat{\mathbf{s}}$ represents the physical constants that correspond to the nominal system model. If we push the nominal model from the initial state to the end of the maneuver we observe an error between the desired final state x^* and the actual simulated final state $\Phi(\mathbf{p}, \hat{\mathbf{s}})$. The nominal parameter set $\mathbf{p}^0$ is a parameter set such that $\Phi(\mathbf{p}^0, \hat{\mathbf{s}}) = x^*$.

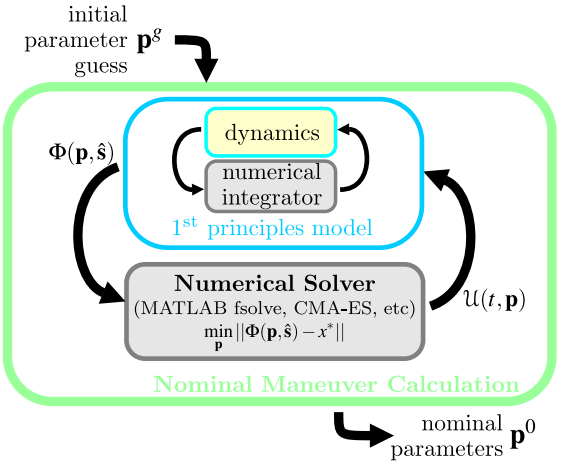

Fig. 3 Procedure for finding the nominal parameter set $\mathbf{p}^0$ from the initial parameter guess $\mathbf{p}^g$. We use a standard numerical integrator that is able to solve the systems of ordinary differential equations describing the nominal dynamics of the system

As shown in Fig. 3, given a well-behaved parameterized input function $\mathcal{U}(t, \mathbf{p})$ and a sufficiently close initial parameter guess $\mathbf{p}^g$, we use a numerical solver to find the nominal parameter set $\mathbf{p}^0$. This provides a nominal form of the maneuver that we now wish to translate to a similar motion with a physical vehicle.

Typically there are more parameters than there are final state errors, leading to an under-defined problem and possibly poor convergence. We use various tools such as optimal control to link and eliminate some free parameters, see Sect. 7.

4.1 Parameter selection

A good choice of variables to include in the parameter set $\mathbf{p}$ is critical for the adaptation to succeed. In this work the selected variables are stage durations and control effort magnitudes—these provide a core set of intuitive "sliders" that control a given input trajectory. Note that the parameter set may also contain more exotic variables such as initial/final state elements (Lupashin and D'Andrea 2011).

The methodology for parameter choice corresponds to formulating policies for policy gradient problems and is outside the scope of this work. It is also highly problem dependent; for the maneuver described in this work, we found the following heuristics useful:

– Each parameter should be linked, to first-order, to at least one measured error variable that is not strongly affected by another parameter.
– The preferred choice for parameters is switch-times or durations. Durations, instead of switch times, work best as this formulation guarantees a fixed switch event ordering.
– In addition, control effort parameters that have direct, linear effect on one or more measured outcome variables.

– Distilling the maneuver to the most concise form helps guarantee that the chosen parameters and errors have clear meaning.

4.2 Parameter constraints

In many cases the parameters are constrained: for example, a time duration parameter is usually constrained to be non-negative, or a thrust parameter should be within the absolute minimum/maximum physical limits. To keep the method description concise, the correction step in the following section does not explicitly take parameter constraints into account.

In practice, parameter constraints are taken into account by applying the following strategy: at each iteration, if a constraint is violated after correction, scale down the correction applied to the current parameter set (see Sect. 5) until the new parameter set is once again valid.

This is a simplistic method of dealing with parameter constraints which, in our experience, works well in practice: disturbances and noise usually result in only transient parameter saturations, if any. A consistent parameter saturation typically means that the maneuver is poorly formulated or is not feasible on the physical system in the given form.

5 Iterative improvement strategy

While the true model of the system is not known, we use the fact that a first-principles model provides the correct overall direction for maneuver-specific corrective action. This is similar to the work described in Kolter and Ng (2009), where it was found that signs alone can provide enough useful information in a gradient matrix to succeed in learning a variety of policy gradient problems.

The optimization of the parameter set using the first-principles model results in an initial parameter set $\mathbf{p}^0$. If the solver succeeded then this parameter set allows the first-principles simulated vehicle to perform the required maneuver, ending exactly in the desired final state x^*.

Given a set of parameters $\mathbf{p}$ and system constants $\mathbf{s}$, let $\mathbf{x} = \Phi(\mathbf{p}, \mathbf{s})$ be the final state of either the nominal system ($\mathbf{s} = \hat{\mathbf{s}}$) or the physical system ($\mathbf{s} = \bar{\mathbf{s}}$) after it is driven with $\mathbf{p}$ via the parameterized input function as described before. Here $\hat{\mathbf{s}}$ is the nominal model of the actual physical constants $\bar{\mathbf{s}}$. It may not be known how many elements $\bar{\mathbf{s}}$ actually has, if it is indeed finite, but the major, first-principles effects in $\bar{\mathbf{s}}$ are properly reflected by $\hat{\mathbf{s}}$.

Assume that $\Phi(\mathbf{p}, \mathbf{s})$ is well-behaved in the neighborhood of the parameter search. In particular, assume that to first order,

$$\Phi(\mathbf{p}, \bar{\mathbf{s}}) = \Phi\big(\mathbf{p}^0, \hat{\mathbf{s}}\big) + \frac{\partial \Phi(\mathbf{p}^0, \hat{\mathbf{s}})}{\partial \mathbf{p}}\big(\mathbf{p} - \mathbf{p}^0\big) + \frac{\partial \Phi(\mathbf{p}^0, \hat{\mathbf{s}})}{\partial \mathbf{s}}(\bar{\mathbf{s}} - \hat{\mathbf{s}}) \tag{1}$$

Here $\Phi(\mathbf{p}^0, \hat{\mathbf{s}})$ is actually x^*, since that is how we selected $\mathbf{p}^0$. Using the nominal model we can also readily calculate the nominal Jacobian $\mathbf{J}$ relating change in parameters to change in the nominal final state:

$$\mathbf{J} = \frac{\partial \Phi(\mathbf{p}^0, \hat{\mathbf{s}})}{\partial \mathbf{p}} \tag{2}$$

In practice, we use a numerical finite-difference method to find $\mathbf{J}$, though more efficient methods exist. We assume that $\mathbf{J}$ is full row rank, meaning that all of the errors can be corrected for using the given parameters.

Let $\mathbf{p}^i$ and $\mathbf{x}^i$ denote the parameter set at iteration i and the resulting final state of the physical system after performing the maneuver. From (1),

$$\mathbf{x}^i = x^* + \mathbf{J}\big(\mathbf{p}^i - \mathbf{p}^0\big) + \mathbf{d} \tag{3}$$

where $\mathbf{d}$ captures modeling error and is unknown. Let $\mathbf{e}^i = \mathbf{x}^i - x^*$ be the error.

We use the following correction strategy:

$$\mathbf{p}^{i+1} = \mathbf{p}^i - \gamma\, \mathbf{C}\mathbf{e}^i \tag{4}$$

where $\mathbf{C}$ is a right matrix inverse of $\mathbf{J}$.

The error then evolves as follows:

$$\mathbf{e}^{i+1} = \mathbf{J}\big(\mathbf{p}^i - \gamma\, \mathbf{C}\mathbf{e}^i - \mathbf{p}^0\big) + \mathbf{d} \tag{5}$$

$$= \mathbf{J}\big(\mathbf{p}^i - \mathbf{p}^0\big) + \mathbf{d} - \gamma\, \mathbf{e}^i \tag{6}$$

$$= (1 - \gamma)\mathbf{e}^i \tag{7}$$

which converges to 0 for $0 < \gamma \leq 1$.

The step size γ provides a tuning parameter for the overall learning algorithm. A high γ should converge faster, but is more susceptible to unrepeatable effects.

In the experiments described later in this work, we use a mixed strategy: $\gamma^i = \max(1/i, 0.1)$, with i beginning at 1, which provides a balance between fast adaptation in the beginning and slow, consistent and noise-resistant adaptation in the long run.

6 Quadrocopters

The maneuvers were implemented and tested on quadrocopters in the Flying Machine Arena. These vehicles are based on the Ascending Technologies Hummingbird platform described in Gurdan et al. (2007), with the wireless communication and central onboard electronics completely replaced by custom-built modules.

The parameter definition and adaptation methodology described previously is not quadrocopter—or aerial vehicle—specific. Quadrocopters were chosen for the experiments because of their agility, robustness and mechanical simplicity.

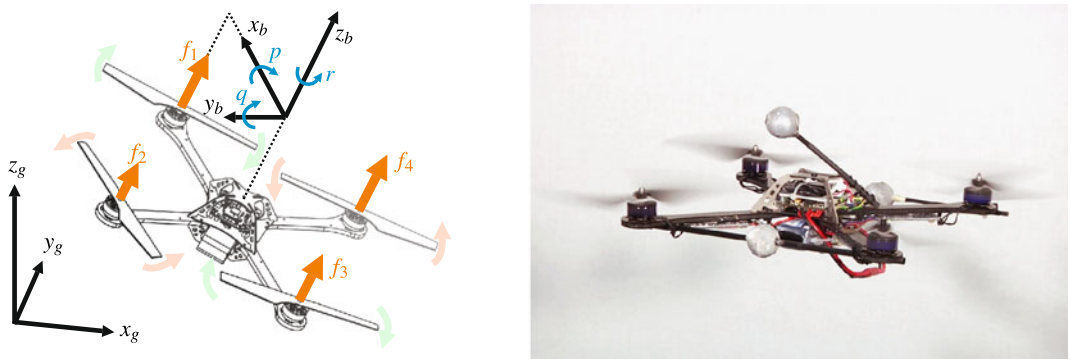

Fig. 4 Global and local coordinate systems, propeller directions, and motor numbering used in this work. For reference, an actual vehicle is shown in hover

At the same time, quadrotor vehicles are perfect as a testing platform for learning algorithms since their dynamics are difficult to model accurately in high-speed flight regimes.

The vehicles used accept four inputs: three body rates $\hat{p}$, $\hat{q}$, $\hat{r}$ and a mass-normalized collective thrust command $\hat{f}$. These inputs are provided either from a wireless data link when flying in full-feedback mode or by an onboard *command generator* when executing an open-loop maneuver. The onboard feedback loop runs at 800 Hz including the command generator, allowing for more precise time granularity than when using commands from the offboard computers (70 Hz). Further details about the experimental setup used in this work are provided in Sect. 10.

An onboard proportional controller uses onboard rate gyros and the given $\hat{p}, \hat{q}, \hat{r}$ commands to calculate desired torques, which are in turn translated to desired differential thrusts by using a nominal vehicle inertia matrix and thrust drag factors. The differential thrusts and the collective thrust are combined by addition to produce desired motor speed commands, which individual motor controllers attempt to follow. The relevant coordinate systems and physical variables are depicted in Fig. 4.

In the following subsections we describe a first-principles, continuous-time model that approximates both the dynamics of the quadrocopter and the behavior of the onboard controllers. Neither time, the inputs, nor the states are discretized, which permits the use of generic finite-difference gradient approximation and numerical optimization routines as required by our method.

6.1 Vehicle dynamics

The model used to generate nominal maneuver parameters and correction matrix and to design the onboard controller is a reduced-order first-principles rigid body dynamics model.

Translational motion in the global frame is given by

$$\begin{bmatrix} \ddot{x}_g \\ \ddot{y}_g \\ \ddot{z}_g \end{bmatrix} = \mathbf{R} \begin{bmatrix} 0 \\ 0 \\ \ddot{z}_b \end{bmatrix} - \begin{bmatrix} 0 \\ 0 \\ g \end{bmatrix} \tag{8}$$

where $\mathbf{R}$ is the rotation matrix from the body frame to the global reference frame and g is acceleration due to gravity.

$\mathbf{R}$ evolves according to the basic definitions of angular body velocities $\boldsymbol{\omega} = (p, q, r)$ (Hughes 1986):

$$\dot{\mathbf{R}} = \mathbf{R} \begin{bmatrix} 0 & -r & q \\ r & 0 & -p \\ -q & p & 0 \end{bmatrix} \tag{9}$$

Special care must be taken when performing numerical integration for $\mathbf{R}$ to remain a valid rotation matrix.

The body rates as well as $\ddot{z}_b$ evolve according to basic kinematics (see, for example, Michael et al. 2010), driven by the current motor thrusts $f_{1..4}$:

$$\mathbf{I}\dot{\boldsymbol{\omega}} = \mathbf{I} \begin{bmatrix} \dot{p} \\ \dot{q} \\ \dot{r} \end{bmatrix} = \begin{bmatrix} l(f_2 - f_4) \\ l(f_3 - f_1) \\ \kappa(f_1 - f_2 + f_3 - f_4) \end{bmatrix} - \boldsymbol{\omega} \times \mathbf{I}\boldsymbol{\omega} \tag{10}$$

$$\ddot{z}_b = (f_1 + f_2 + f_3 + f_4)/m \tag{11}$$

where $\mathbf{I}$ is the inertia matrix of the vehicle (assumed diagonal in this work), l is the vehicle center to rotor distance, κ is an experimentally determined constant and m is the mass of the vehicle. The values of these and other parameters used in this work are listed in Table 1.

We model each motor as a first-order system with constraints $f_{min} \leq f_i \leq f_{max}$. From experiments we have found that the motors are quicker to produce more thrust (spinning up) than producing less thrust (spinning down). We use sensorless brushless motors and Ascending Technologies speed-control motor controllers which do not perform active braking, likely leading to this asymmetry:

$$\dot{f}_i = P_f(\tilde{f}_i - f_i) \tag{12}$$

Table 1 Quadrocopter model parameter values

	Description	Value
m	vehicle mass	0.468 kg
l	motor-center length	0.17 m
$\mathbf{I}_{xx}, \mathbf{I}_{yy}$	inertia about x_b, y_b (est.)	0.0023 kg m^2
$\mathbf{I}_{zz}$	inertia about z_b (est.)	0.0046 kg m^2
$P_{f,up}$	thrust increase speed (est.)	80 s^{-1}
$P_{f,dn}$	thrust decrease speed (est.)	60 s^{-1}
f_{min}	min rotor thrust	0.08 N
f_{max}	max rotor thrust	2.8 N
$\underline{\beta}$	reduced min collective accel.	3.92 m/s^2
$\overline{\beta}$	reduced max collective accel.	21.58 m/s^2
κ	thrust-to-drag constant (est.)	0.016 m
P_p, P_q	angular rate feedback gain (p, q)	100 s^{-1}
P_r	angular rate feedback gain (r)	10 s^{-1}

where $\tilde{f}_i$ is the desired thrust command from the onboard feedback controller as explained in the following section and $P_f = P_{f,up}$ when $\tilde{f}_i > f_i$ and $P_f = P_{f,dn}$ otherwise.

For purposes of defining final attitude errors in the following sections we use a Z–Y–X Euler attitude parameterization ϕ, θ, ψ, which can be extracted from $\mathbf{R}$ (Diebel 2006):

$$\begin{bmatrix}\phi\\ \theta\\ \psi\end{bmatrix} = \begin{bmatrix}\text{atan2}(r_{23}, r_{33})\\ -\text{asin}(r_{13})\\ \text{atan2}(r_{12}, r_{11})\end{bmatrix}, \qquad \mathbf{R} = \begin{bmatrix}r_{11} & r_{12} & r_{13}\\ r_{21} & r_{22} & r_{23}\\ r_{31} & r_{32} & r_{33}\end{bmatrix} \tag{13}$$

6.2 Onboard feedback controller

The purpose of the onboard controller is to cancel some of the unmodeled or unpredictable effects by doing high-rate feedback on angular body rates using the rate gyros. It consists of three separate proportional controllers for each of the body axes that calculate desired angular accelerations $\tilde{a}_p, \tilde{a}_q, \tilde{a}_r$ from current gyro readings p, q, r and desired angular rates $\hat{p}, \hat{q}, \hat{r}$:

$$\tilde{a}_p = P_p(\hat{p} - p), \tag{14}$$

$$\tilde{a}_q = P_q(\hat{q} - q), \tag{15}$$

$$\tilde{a}_r = P_r(\hat{r} - r) \tag{16}$$

where $P_{p,q,r}$ are proportional gains.

These desired angular accelerations are then converted to individual motors commands by inverting (10):

$$\tilde{f}_1 = (\hat{f} + \tilde{\mu}_r/\kappa - 2\tilde{\mu}_q/l)/4, \tag{17}$$

$$\tilde{f}_2 = (\hat{f} - \tilde{\mu}_r/\kappa + 2\tilde{\mu}_p/l)/4, \tag{18}$$

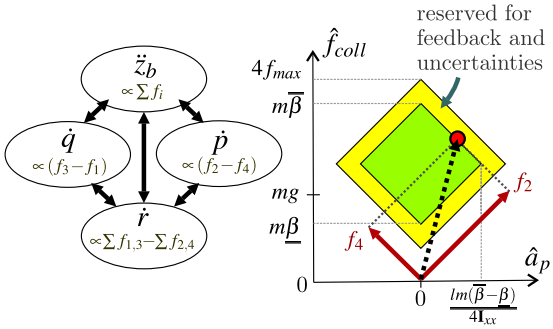

Fig. 5 Interaction of quadrocopter acceleration inputs due to constraints and sample two-input control input envelope, used for a quadrocopter performing a flip in roll. The control envelope pictured on the right assumes that $f_1 = f_3 = (f_2 + f_4)/2$. The *green area* is the *reduced control envelope* used in this work for the multi-flip maneuver

$$\tilde{f}_3 = (\hat{f} + \tilde{\mu}_r/\kappa + 2\tilde{\mu}_q/l)/4, \tag{19}$$

$$\tilde{f}_4 = (\hat{f} - \tilde{\mu}_r/\kappa - 2\tilde{\mu}_p/l)/4 \tag{20}$$

where $\hat{f}$ is a collective thrust command, and $(\tilde{\mu}_p, \tilde{\mu}_q, \tilde{\mu}_r)$ are desired moments that are calculated from desired angular accelerations $(\tilde{a}_p, \tilde{a}_q, \tilde{a}_r)$ and the current body rates $\boldsymbol{\omega}$:

$$\begin{bmatrix}\tilde{\mu}_p\\ \tilde{\mu}_q\\ \tilde{\mu}_r\end{bmatrix} = \mathbf{I}\left(\begin{bmatrix}\tilde{a}_p\\ \tilde{a}_q\\ \tilde{a}_r\end{bmatrix} + \mathbf{I}^{-1}\boldsymbol{\omega} \times \mathbf{I}\boldsymbol{\omega}\right) \tag{21}$$

7 Tools for maneuver design

We have found several well-known tools and concepts from control theory to be useful for both creating new maneuvers and for selecting parameters for applying the described method to learn or adapt existing maneuvers.

7.1 Time-optimal control

For this method to work effectively we need to select as few parameters to be changed as possible while maintaining intuitiveness and enough freedom for the maneuver to adapt. We found concepts from time-optimal control such as bang-bang control very useful in this regard.

We first consider the interaction of inputs due to saturation and the resulting feasible control envelope of the vehicle. The control envelope may be defined at several levels. Since there is an onboard controller following angular rate commands, we choose to consider angular accelerations, which are subject to linear saturation constraints between each other and the feasible collective acceleration of the vehicle, as shown in the left part of Fig. 5.

For bang-bang control inputs, shown in the past to be very close to time-optimal (Purwin and D'Andrea 2011), we

wish all of the control inputs to lie on the edges of the feasible control envelope. At the same time, the onboard feedback controllers need some reserved control effort to work, and there may be significant modeling inaccuracies, so we choose to reduce the control envelope by some margin. For a bang-bang control strategy all control inputs should then lie on the edges of this reduced control envelope.

In the examples included in this work, we use bang-bang control to reduce the number of free parameters when designing the multi-flip maneuver in Sect. 8. To simplify discussion we now focus on a quadrocopter operating in the vertical Y–Z plane. The resulting control envelope is pictured in the right part of Fig. 5.

We can derive the coupling between angular acceleration and collective acceleration constraints by inspecting the interplay of the constraints of the underlying motor thrusts. Simplifying (10) by focusing on f_2 and f_4 by setting $f_1 = f_3 = (f_2 + f_4)/2$ and assuming that $\mathbf{I}$ has no off-diagonal terms and ignoring the higher order effects,

$$\ddot{z}_b = 2(f_2 + f_4)/m, \tag{22}$$

$$\dot{p} = l(f_2 - f_4)/\mathbf{I}_{xx} \tag{23}$$

If we introduce reduced-envelope minimum and maximum mass-normalized motor thrusts $\underline{\beta}$ and $\overline{\beta}$ such that

$$4f_{min} < m\underline{\beta} \leq 4f_i \leq m\overline{\beta} < 4f_{max} \tag{24}$$

and wish to use only bang-bang commands, the collective acceleration is now a function of the angular acceleration:

$$\hat{f}_{coll,high} = m\overline{\beta} - 2\left|\hat{a}_p(t)\right|\mathbf{I}_{xx}/l \tag{25}$$

$$\hat{f}_{coll,low} = m\underline{\beta} + 2\left|\hat{a}_p(t)\right|\mathbf{I}_{xx}/l \tag{26}$$

$$\left|\hat{a}_p(t)\right| \leq lm(\overline{\beta} - \underline{\beta})/4\mathbf{I}_{xx} \tag{27}$$

and vice versa:

$$\left|\hat{a}_p(t)\right| = \frac{l}{2\mathbf{I}_{xx}}\left(\frac{m(\overline{\beta} - \underline{\beta})}{2} - \left|\hat{f} - \frac{m(\overline{\beta} + \underline{\beta})}{2}\right|\right) \tag{28}$$

$$\underline{\beta} \leq \hat{f} \leq \overline{\beta} \tag{29}$$

Time-optimal control has also been used in sophisticated ways to find nominal near-time-optimal trajectories for maneuvers such as fast translation (Ritz et al. 2011). Due to the extremes involved these maneuvers typically do not perform well in the physical world in their nominal form, but are perfect as a starting point for an adaptive parameterized maneuver, as discussed in Sect. 9.

7.2 Hidden system states

The true model of the vehicle is considered unknown in this paper; furthermore, even the number of states in the model is unknown. For example, one could easily imagine including thousands of variables in the model state to accurately simulate blade bending or the airflow around the vehicle during a fast motion to get more accurate dynamic thrust predictions.

In order to have better experimental repeatability and to have the learned maneuver be closer to the desired outcome, it is important to add ramps to transition from aerobatic control to the more tame, typically linear controller that takes over once the vehicle completes the maneuver. We use a fixed-duration ramp-down for thrust to hover in all cases and a fixed-duration ramp-down for angle rate commands when the maneuver is defined in terms of angular rates and not accelerations. The ramp-down is simply an envelope constraining the appropriate input to approach the nominal hover value as the maneuver nears its end.

We found that including these ramp-downs in the maneuvers vastly improves the visual quality of the motion while having only minimal effect on the initial iterations or the learning. Note that the ramp-downs are included in the maneuver profile throughout all simulated/nominal and experimental runs.

8 Sample application: multi-flips

Aerobatics typically require pushing a vehicle to its physical limits to reliably perform an awe-inspiring maneuver. Autonomous aerial vehicles have been shown effectively mimicking the performance of expert human pilots (Hoffmann et al. 2011; Gerig 2008), but we may also seek to perform arbitrary new maneuvers that we can conceptualize but not demonstrate, such as flips at very high rates.

To demonstrate the proposed method, we wish to perform a fast multi-flip from hover and ending in hover. The vehicle should fly up, rotate N times around one of its axes, and come back down, ending in the same spot in a static hover state. We do not have a demonstration of this maneuver and we do not have a trajectory to follow.

This experiment is described in detail in Lupashin et al. (2010) with extensions in Lupashin and D'Andrea (2011), so we provide a shorter treatment here. The maneuver is defined by fixed user-defined parameters $\mathbf{c}_N$ and $\mathbf{c}_{p_{max}}$ that set the number of flips to be performed and the maximum angular rate to follow, respectively.

The relevant state elements, initial and desired final state are as follows:

$$x = (y, \dot{y}, z, \dot{z}, \theta), \tag{30}$$

$$x(0) = (0, 0, 0, 0, 0), \tag{31}$$

$$x^* = (0, 0, 0, 0, 2\pi\mathbf{c}_N) \tag{32}$$

For this maneuver we define the reference trajectory in the form of three angular accelerations $\hat{a}_p(t), \hat{a}_q(t), \hat{a}_r(t)$

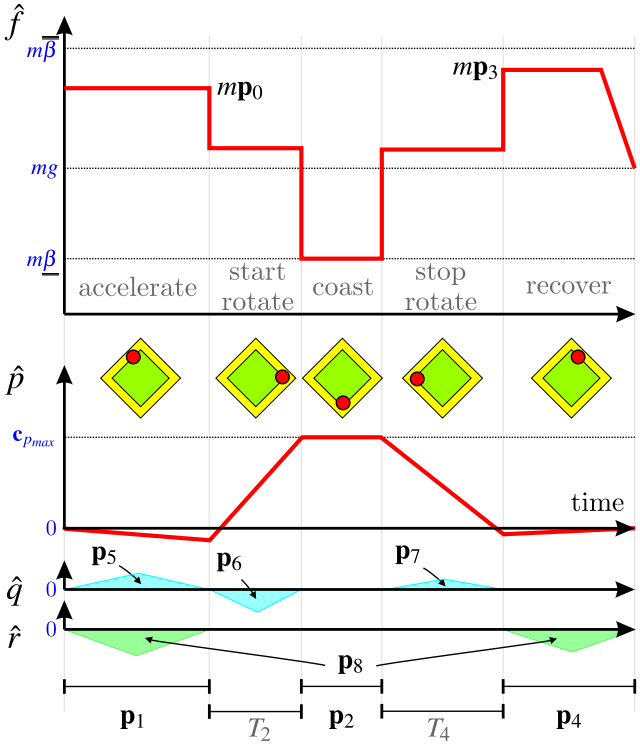

Fig. 6 Parameterized input function used for the 2D (and, with the lower parts, 3D) adaptive hover-to-hover multiflips. The *red dots* in the *diamonds* show where approximately, the roll and collective thrust inputs lie on the boundary of the reduced control envelope for each stage. The 3D adaptive motion parameters $\mathbf{p}_{5..8}$ refer to the areas of the marked *triangles*, with $\mathbf{p}_8$ referring to the sum of the areas of the *two triangles* (i.e. the total yaw angle correction applied during maneuver)

and a collective thrust $\hat{f}(t)$. Since the onboard controller follows angular rates and not angular accelerations, we simply integrate the angular accelerations from 0 to calculate the reference body rate commands.

We split the maneuver into five stages: *accelerate*, *start rotate*, *coast*, *stop rotate*, and *recover*. Each stage is defined by a duration, a constant angular acceleration command, and a constant collective thrust command. Using constraints on the end conditions, the assumption that we will reach the coast phase, and the requirement to always send commands lying on the edge of the reduced control envelope from Sect. 7.1, we can reduce this maneuver to being fully described by five parameters. This is perfect since, in the 2D case, the error vector has five members: $(y, \dot{y}, z, \dot{z}, \theta)$, leading to a square correction matrix and a well-behaved problem.

The parameterized input function used for the 2D and 3D adaptive multi-flips is pictured in Fig. 6.

For the 2D case, we set $\hat{a}_q = \hat{a}_r = 0$. Parameters $\mathbf{p}_1, \mathbf{p}_2$ and $\mathbf{p}_4$ are defined as the durations of the *accelerate*, *coast*, and *recover* stages, respectively. Parameters $\mathbf{p}_0$ and $\mathbf{p}_3$ are mass-normalized collective thrust commands during the *accelerate* and *recover* stages. The roll acceleration and collective thrust commands are:

$$\hat{a}_p(t) = \begin{cases} \text{accelerate} & -ml(\overline{\beta} - \mathbf{p}_0)/4\mathbf{I}_{xx} \\ \text{start rotate} & ml(\overline{\beta} - \underline{\beta})/4\mathbf{I}_{xx} \\ \text{coast} & 0 \\ \text{stop rotate} & -ml(\overline{\beta} - \underline{\beta})/4\mathbf{I}_{xx} \\ \text{recover} & ml(\overline{\beta} - \mathbf{p}_3)/4\mathbf{I}_{xx}, \end{cases} \tag{33}$$

$$\hat{f}(t) = \begin{cases} \text{coast} & m\underline{\beta} \\ \text{otherwise} & m\overline{\beta} - 2|\hat{a}_p(t)|\mathbf{I}_{xx}/l \end{cases} \tag{34}$$

The other missing variables can be solved for algebraically:

$$T_2 = (\mathbf{c}_{p_{max}} - \mathbf{p}_1 \hat{a}_{p,accelerate})/\hat{a}_{p,startRotate}, \tag{35}$$

$$T_4 = (\mathbf{c}_{p_{max}} + \mathbf{p}_4 \hat{a}_{p,stopRotate})/\hat{a}_{p,stopRotate} \tag{36}$$

8.1 Initial parameter guess

The initial parameter guess $\mathbf{p}^g$ given to the numeric solver to find the nominal parameter set $\mathbf{p}^0$ is quite important for this maneuver since it may result in finding the wrong minima, such as a double flip instead of a desired triple flip. We make a conservative estimate that about 90 % of the thrust should be used for vertical acceleration during *accelerate* and *recover* stages and use back-of-the-envelope algebraic analysis to calculate $\mathbf{p}^g$ (see Lupashin et al. 2010, for more details):

$$\mathbf{p}_0^g = \mathbf{p}_3^g = 0.9\overline{\beta}, \tag{37}$$

$$\mathbf{p}_1^g = \mathbf{p}_4^g = \frac{g(T_2 + T_3 + T_4)}{2\mathbf{p}_0^g}, \tag{38}$$

$$\mathbf{p}_2^g = \frac{2\pi \mathbf{c}_N}{\mathbf{c}_{p_{max}}} - \frac{\mathbf{c}_{p_{max}}}{\hat{a}_{p,startRotate}} \tag{39}$$

8.2 Extension to 3D

After running the 2D adaptive maneuver we found that for all but perfectly-calibrated, perfectly-balanced new vehicles, the other degrees of freedom would drift significantly during the maneuver. We extend the maneuver by first considering a more inclusive initial and desired final state definition:

$$x = (y, \dot{y}, z, \dot{z}, \theta, x, \dot{x}, \phi, \psi), \tag{40}$$

$$x(0) = (0, 0, 0, 0, 0, 0, 0, 0, 0), \tag{41}$$

$$x^* = (0, 0, 0, 0, 2\pi \mathbf{c}_N, 0, 0, 0, 0) \tag{42}$$

The other degrees of freedom are controlled by learning cumulative angular offsets during the various "up-facing"

parts of the maneuver. For this we define helper functions:

$$\mathbb{F}_1(A, t_s, t_\Delta) = \frac{A\ \mathrm{sgn}((2t_s + t_\Delta)/2 - t)}{t_\Delta^2}, \tag{43}$$

$$\mathbb{F}_2(A, t_s, t_\Delta, t_e) = \frac{A\ \mathrm{sgn}((t_s + t_e)/2 - t)}{t_\Delta^2} \tag{44}$$

We note that we have four new members in the error vector $\mathbf{e}$. The lateral inputs are then defined using four parameters as follows:

$$\hat{a}_q(t) = \begin{cases} \text{accelerate} & \mathbb{F}_1(\mathbf{p}_5, 0, \mathbf{p}_1) \\ \text{start rotate} & \mathbb{F}_1(\mathbf{p}_6, \mathbf{p}_1, T_2) \\ \text{stop rotate} & \mathbb{F}_1(\mathbf{p}_7, \mathbf{p}_1 + T_2 + \mathbf{p}_2, T_4) \\ \text{otherwise} & 0, \end{cases} \tag{45}$$

$$\hat{a}_r(t) = \begin{cases} \text{accelerate} & \mathbb{F}_2(\mathbf{p}_8, 0, \mathbf{p}_1 + \mathbf{p}_4, \mathbf{p}_1) \\ \text{recover} & \mathbb{F}_2(\mathbf{p}_8, t_{end} - \mathbf{p}_4, \mathbf{p}_1 + \mathbf{p}_4, t_{end}) \\ \text{otherwise} & 0 \end{cases} \tag{46}$$

where $t_{end} = \mathbf{p}_1 + T_2 + \mathbf{p}_2 + T_4 + \mathbf{p}_4$.

To extend $\mathbf{p}^g$, we simply observe that nominally no corrective lateral action should be required, so the initial parameter guess for the new parameters is 0.

9 Sample application: translation motion benchmarking

In Ritz et al. (2011), Pontryagin's minimum principle was used to numerically compute time-optimal maneuvers for quadrocopters moving from hover at one point to another. The numerical method was applied to horizontal translation, vertical translation, and arbitrary hover-to-hover vertical plane movement. Together these motions provide a time benchmark for a large class of common quadrocopter motions—a useful tool for evaluating any existing or future controllers.

The nominal maneuver computation uses the following constants: the maximum angular rate of the vehicle $\mathbf{c}_{p_{max}}$ and the minimum and maximum limits for the collective thrust command $\hat{f}$. For the motion benchmarking maneuvers in this work, $\mathbf{c}_{p_{max}} = 10$ rad/s. The thrust limits used for the adaptive maneuver are from the reduced envelope described in Sect. 7, reduced even further to allow the vehicle to follow steps in the desired angular rate:

$$m\big(\overline{\beta} + 0.05(\overline{\beta} - \underline{\beta})\big) \le \hat{f} \le m\big(\overline{\beta} + 0.05(\underline{\beta} - \overline{\beta})\big) \tag{47}$$

Due to the time-optimality criterion, the maneuvers result in bang-bang inputs with occasional singular arcs. The maneuvers are defined in terms of collective thrust and an angular rate, both of which are assumed by the benchmarking method to switch instantaneously to the commanded values. The real vehicle is not able to physically follow such commands, and the extreme velocities and input switches result in significant aerodynamic and other unmodeled effects dominating the resulting motion. Because of this, the maneuvers resulted in large errors in trying to reach a target point (see Fig. 12 for an example).

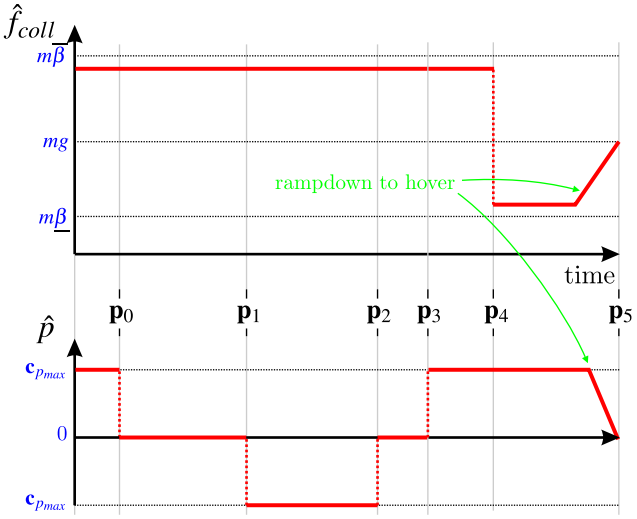

Fig. 7 The parameterized input trajectory for a diagonal translation maneuver from $(0, 0)$ to $(5, 5)$. The resulting nominal and learned trajectories can be seen in Fig. 12. Counter-intuitively, the quadrocopter actually flips over during the maneuver to achieve the fast maneuver completion time

The motion benchmarking algorithm takes a desired (y, z) coordinate pair as an input and produces as output a nominal vehicle and input trajectory. The input trajectory is in the form of bang-bang style switches with occasional singular arcs. For example, for a translation maneuver from $(0, 0)$ to $(5, 5)$, the nominal input trajectory predicted by the motion benchmarking algorithm is something similar to Fig. 7, with a single switch in collective thrust and four switches in angular rates (the ramp-downs were added to the motion when transferring it to the physical vehicle).

For different translations the number of switches and the type of singular arcs varies, but we always selected the switch times as the parameters to be learned—for the $(5, 5)$ diagonal translational motion, the parameters are pictured in Fig. 7.

Since the motion benchmarking model uses a very simple, instantaneous angle-rate model, the nominal switch times then form the initial parameter guess $\mathbf{p}^g$. We then use the usual procedure to find $\mathbf{p}^0$ and the correction matrix $\mathbf{C}$.

For the translational maneuvers there are usually more switch times than error states, so we arbitrarily used a least-squares style pseudo-inverse of $\mathbf{J}$ to find the correction matrix:

$$\mathbf{C} = \mathbf{J}^T\left(\mathbf{J}\mathbf{J}^T\right)^{-1} \tag{48}$$

The rest of the algorithm was applied as usual. Several translation maneuvers were learned; in this work we show results for $(5, 5)$ diagonal translation. Please see Ritz et al. (2011) for further learned results.

10 Experiments

10.1 The testbed

The algorithm was implemented in the ETH Flying Machine Arena (FMA), a dedicated testbed for motion control research. More information about the FMA can be found in Lupashin et al. (2010) and at the FMA website;[1] we provide a brief overview here of the relevant details.

At the top level, the FMA is organized similarly to the MIT Raven testbed (How et al. 2008). It uses an 8-camera motion capture system, running at 200 Hz and providing mm- and degree- accurate position and rotation measurements for any appropriately calibrated rigid bodies. In this work, the motion capture system enables the vehicle to set up in the initial state and provides final state error measurements so that the maneuvers can be improved.

Several off-the-shelf computers serve as development and offboard computation platforms for the FMA. They receive the motion capture data, run various estimation algorithms to filter the data, and run standard controllers that enable flight with full sensor feedback. Note that the full closed-loop communication latency of the FMA, from observing a marker on a vehicle to the vehicle receiving wireless commands relating to that observation, is between 20 and 50 ms. This provides a pragmatic reason to use the described methodology to learn fast maneuvers such as flips: like for fast human motions such as flips in gymnastics or ball hits in tennis, full sensory feedback is too late (or the time delay variance too great) to be of direct use during the maneuver. For example, a flip at 1800°/s would end up around 80 degrees off-center if done in full feedback.

There are two ways to communicate with the quadrocopters: an 802.11b system for bidirectional communication such as diagnostics and onboard sensor feedback and parameter read/writes and a unidirectional broadcast FHSS 2.4 GHz system for sending commands to the vehicle. The latter is used to trigger the maneuver. Commands are sent at 70 Hz with a typical drop rate of 5 %.

10.2 Maneuver implementation

The maneuvers are stored onboard the quadrocopter as piecewise functions, represented as a list of switches. Each switch consists of a switch time, a switch type (for example, commanded roll rate $\hat{p}$), and a value. This allows for a great variety of possible functions that can be sampled at low computational cost and at the full onboard control rate. Using commands generated onboard during the maneuvers allows for much better granularity and repeatability than with the standard wireless command interface that suffers from dropouts, variable time delays, and low time resolution.

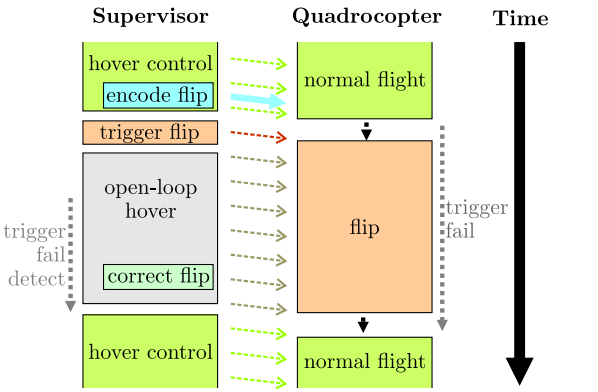

Fig. 8 The process for a single iteration of learning a flip (the same exact process is used for any other maneuvers). The *middle arrows* represent commands sent from the offboard computers (*left side*) to the vehicle (*right side*). Some details such as the vehicle acknowledging receiving a new maneuver definition are omitted for clarity

We call the program running offboard, managing the learning process and also flying the vehicle between trials the *supervisor*. The supervisor nominally runs standard cascaded PD controllers to bring the vehicle to the starting position.

The steps involved in performing a single iteration maneuver learning are depicted in Fig. 8. For each iteration, the parameterized maneuver input function is used to translate the current parameter set into a piecewise onboard reference functions, sent to the quadrocopter by the supervisor before the maneuver is triggered. A single "trigger" command is then sent over the command link to trigger the maneuver and the offboard controller begins to collect state data. The time delay between the trigger command and the quadrocopter can be measured so the supervisor is able to estimate accurately when the maneuver finishes, if the trigger is received.

At that point the supervisor stops collecting state data, commands the vehicle to hover and return to the starting position, and attempts to accurately determine the final state error for that iteration of the maneuver.

The final state error is then multiplied by the correction matrix $\mathbf{C}$ and by the step size γ. The supervisor then checks if the correction will violate parameter constraints; if it does, the correction is scaled down until the constraints are not violated. The correction is applied, the maneuver is regenerated and reuploaded, and the process is repeated.

[1] www.FlyingMachineArena.org.

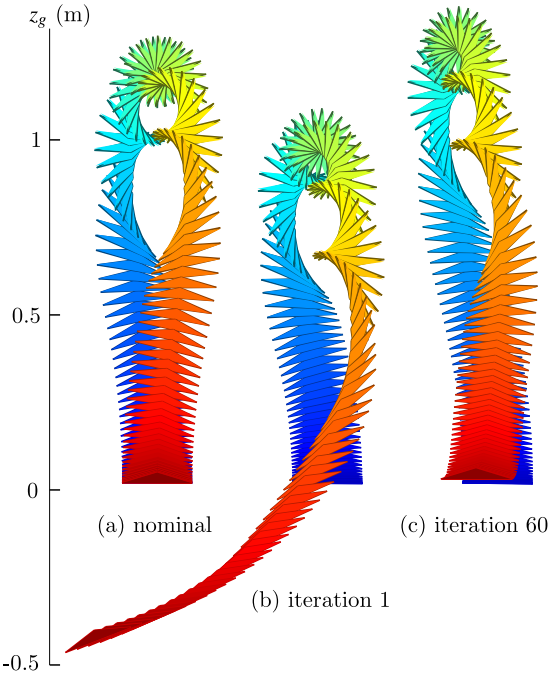

Fig. 9 Triple flips designed and learned using the described method. Parameters: $\mathbf{c}_N = 3$, $\mathbf{c}_{p_{max}} = 1800°/s$. All trajectories shown sampled at 100 Hz. The width of the triangles corresponds approximately to actual width of the vehicle. The figure shows: (**a**) The nominal trajectory. (**b**) Vehicle executing the nominal maneuver (first iteration) before any adaptation. (**c**) Vehicle executing learned maneuver after 60 iterations

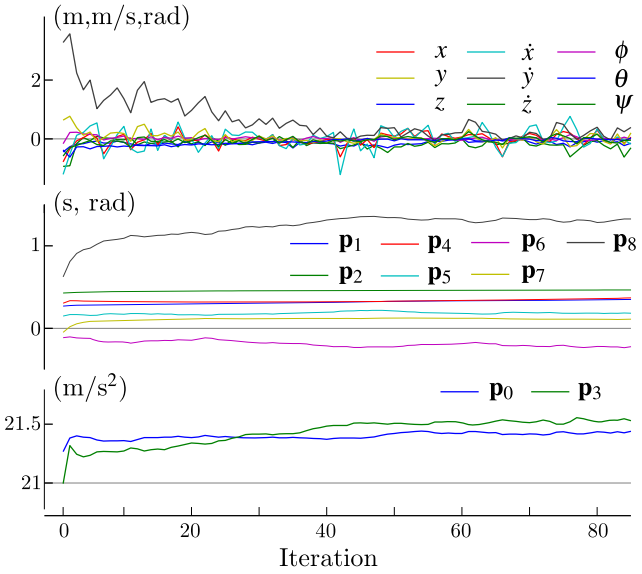

Fig. 10 Evolution of final state errors and parameters for a triple 1800°/s hover-to-hover 3D adaptive flip. This data was collected from a single quadrocopter flight. Detailed side views of iterations 1 and 60 are shown in Fig. 9. Note that the parameters keep changing slightly after an initial period of rapid adjustment. We believe that this is due to subtle changes to system dynamics over time, such as propeller wear, shifting battery characteristics, etc.

For maximum time accuracy, exactly one trigger command is sent to the vehicle to trigger the maneuver. After sending the trigger command the controller sends placeholder open-loop hover commands that are ignored by the vehicle if it executes the maneuver but keep the vehicle airborne in case it does not actually receive a trigger. A timeout detects trigger failure conditions, in which case the process is reset, no correction is applied, and the full iteration is attempted again.

10.3 Flips

Figure 9 shows the Y–Z side view of triple, 1800°/s flips, in the nominal form, before learning, and after a number of iterations. Figure 10 shows the evolution of the maneuver as the parameters are adapted from experiments. There are strong unrepeatable effects, reflected as noise of the final state errors. Note that the 3D adaptation learns a significant offset in $\mathbf{p}_8$, compensating for significant unmodeled yaw dynamics that occur during the flip.

Figure 11 shows a single 800°/s flip for comparison. Note that the slower single flip is much more symmetrical than the fast triple flip, showing how the various unmodeled effects dominate the fast triple flip. Also, surprisingly, the vehicle, after learning, needs less height than nominal for the single flip, while it needs more height than nominal for the triple flip. This was an unexpected result and shows the algorithm being able to cope with a variety of different effects even though the base maneuver description is the same and even though the correction matrices for the two maneuvers are very close.

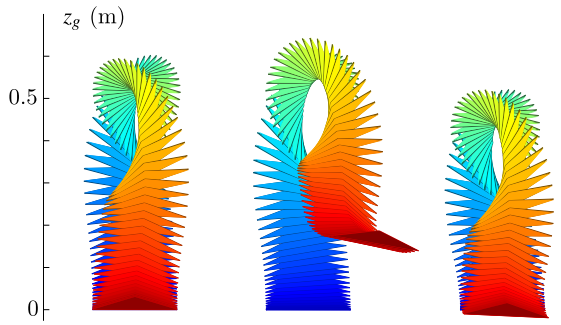

Fig. 11 Nominal trajectory, first attempt, and iteration 17 for a 800°/s single flip. Interestingly, in contrast to Fig. 9, the learned maneuver takes less height relative to the nominal maneuver, even though the design and adaptation method is identical for both maneuvers

10.4 Motion benchmarking

Figure 12 shows the result of applying the described adaptation method to a $(0, 0)$ to $(5, 5)$ time-optimal translation maneuver. The benchmarking algorithm predicts a duration of 1.38 seconds, the nominal maneuver ($\mathbf{p}^0$) is 1.53 seconds but on the physical system ends up about halfway away from

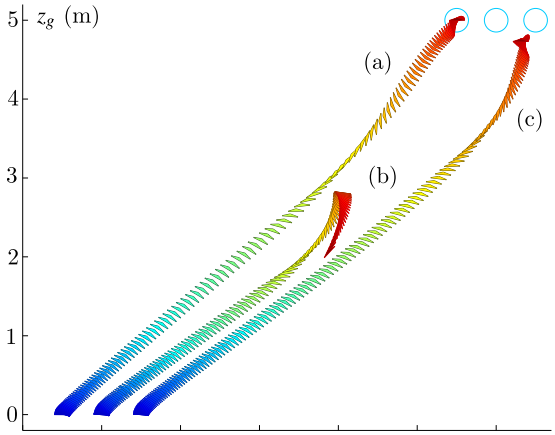

Fig. 12 (**a**) Nominal trajectory, (**b**) initial attempt, and (**c**) iteration 25 of a near-time-optimal benchmark translational motion. The plots are offset by 0.5 m for clarity. The blue circles mark the ideal final states. The learned motion succeeds in getting close to the desired position and takes 1.64 seconds versus the benchmarking algorithm nominal prediction of 1.38 seconds

the target point, while the duration of the final adapted maneuver is 1.64 seconds. The benchmarking algorithm predicts the shortest duration because of the assumption that arbitrary angular rate jumps can be performed by the vehicle without penalty. The nominal model is a more accurate representation, leading to a longer time, but still ignores many effects, leading to a different (in this case longer) physically demonstrated maneuver duration.

The physical demonstration of the benchmarking motion provides a grounding and a reference point for evaluating the performance of other controllers.

11 Discussion and conclusion

The presented method has been shown to effectively learn to compensate for unmodeled repeatable disturbances for fast, high-performance quadrocopter maneuvers. To keep the approach as straightforward as possible, we have made various assumptions such as the initial state being exactly the nominal initial state, that non-systematic errors are minimal, that parameter constraints do not affect convergence for well-behaved maneuvers, and that the second-order effects of parameter change with respect to final state errors are negligible.

These assumptions appear to hold for the demonstrated maneuvers but it is worth reiterating that the proposed method is described here as a quick, lightweight tool to try when implementing a motion on a physical system. It may not work in all cases, but it should be quick and easy enough to try so that little is lost in the event that it does not converge for a given motion.

More sophisticated methods can be used to make the method more reliable. For example, the nominal model can also be used to predict and to compensate for non-ideal initial conditions. In addition we can mix in feedback to compensate for some of the non-systematic disturbances. A challenge going forward with this methodology will be to develop these extensions, but in such a way that the algorithm remains as straightforward and generic as possible.

We have also avoided using a more complex parameter update methods. Learning the parameterized maneuvers can be seen as estimating a set of static variables, each iteration providing an observation. A Kalman filter with state constraints (Simon and Chia 2002) can be used, for example, for this task, and it can readily take advantage of other readily available information such as the relative noise levels on the final variables.

This work focused on short-duration, high-performance, time-optimal-inspired open-loop maneuvers as an application of the proposed adaptation method. This algorithm can also be readily applied to other types of maneuvers, as long as one takes care to select few, intuitively strong parameters to take full advantage of the directness of this approach.

To conclude, we have presented a methodology and learning algorithm for designing parameterized open-loop maneuvers and applied it to quadrocopters. We presented some basic tools to help in designing the maneuver, reducing the number of free parameters, and getting good results on the physical system. As demonstrations we have presented fast hover-to-hover multi-flips and how the method was applied to a theoretical numeric motion benchmarking project to demonstrate fast translation on a physical quadrocopter. The described method is simple and may not be the best choice for all situations, but we hope that it can be a useful tool for working with fast, difficult-to-model systems.

Acknowledgements We thank Markus Hehn and Angela Schoellig for their contributions to the Flying Machine Arena and for the fruitful discussions about all things flying and falling.

References

Abbeel, P., Coates, A., & Ng, A. Y. (2010). Autonomous helicopter aerobatics through apprenticeship learning. *The International Journal of Robotics Research*, *29*, 1608–1639.

Diebel, J. (2006). Representing attitude: Euler angles, unit quaternions, and rotation vectors.

Gerig, M. B. (2008). *Modeling, guidance, and control of aerobatic maneuvers of an autonomous helicopter*. Ph.D. thesis, ETH Zurich, No. 17805.

Gillula, J. H., Huang, H., Vitus, M. P., & Tomlin, C. J. (2009). Design and analysis of hybrid systems, with applications to robotic aerial vehicles. In *International symposium of robotics research*, 2009.

Gurdan, D., Stumpf, J., Achtelik, M., Doth, K. M., Hirzinger, G., & Rus, D. (2007). Energy-efficient autonomous four-rotor flying robot controlled at 1 kHz. In *IEEE international conference on robotics and automation, 2007* (pp. 361–366).

Hoffmann, G. M., Huang, H., Waslander, S. L., & Tomlin, C. J. (2011). Precision flight control for a multi-vehicle quadrotor helicopter testbed. *Control Engineering Practice*, *19*(9), 1023–1036.

How, J., Bethke, B., Frank, A., Dale, D., & Vian, J. (2008). Real-time indoor autonomous vehicle test environment. *IEEE Control Systems Magazine*, *28*(2), 51–64.

Hughes, P. C. (1986). *Spacecraft attitude dynamics*. New York: Wiley. ISBN 0-471-81842-9.

Kolter, J. Z., & Ng, A. Y. (2009). Policy search via the signed derivative. In *Robotics: science and systems*.

Leishman, J. G. (2006). *Principles of helicopter aerodynamics* (2nd edn.). Cambridge: Cambridge University Press.

Lupashin, S., & D'Andrea, R. (2011). Adaptive open-loop aerobatic maneuvers for quadrocopters. In *IFAC world congress*.

Lupashin, S., Schöllig, A., Sherback, M., & D'Andrea, R. (2010). A simple learning strategy for high-speed quadrocopter multi-flips. In *IEEE international conference on robotics and automation (ICRA), 2010* (pp. 1642–1648).

Mellinger, D., Michael, N., & Kumar, V. (2010). Trajectory generation and control for precise aggressive maneuvers with quadrotors. In *Int. symposium on experimental robotics*.

Michael, N., Mellinger, D., Lindsey, Q., & Kumar, V. (2010). The GRASP multiple micro-UAV testbed. *IEEE Robotics & Automation Magazine*, *17*(3), 56–65.

Purwin, O., & D'Andrea, R. (2011). Performing and extending aggressive maneuvers using iterative learning control. *Robotics and Autonomous Systems*, *59*, 1–11.

Ritz, R., Hehn, M., Lupashin, S., & D'Andrea, R. (2011). Quadrotor performance benchmarking using optimal control. In *IEEE/RSJ international conference on intelligent robots and systems* (pp. 5179–5186).

Simon, D., & Chia, T. (2002). Kalman filtering with state equality constraints. *IEEE Transactions on Aerospace and Electronic Systems*, *39*, 128–136.

Wolpert, D., & Flanagan, J. (2010). Motor learning. *Current Biology*, *20*(11), R467–472.

Wolpert, D., Ghahramani, Z., & Flanagan, J. (2001). Perspectives and problems in motor learning. *Trends in Cognitive Sciences*, *5*(11), 487–494.

Sergei Lupashin received the B.Sc. degree in Electrical and Computer Engineering from Cornell University in 2006 and the M.S. degree in Mechanical Engineering from ETH Zurich in 2010. He participated in the RoboCup, DARPA Grand Challenge, and DARPA Urban challenge teams at Cornell. He is currently a Ph.D. student at the Institute for Dynamic Systems and Control, ETH Zurich, developing the Flying Machine Arena testbed and doing research on multi-agent autonomous systems.

Raffaello D'Andrea received the B.Sc. degree in Engineering Science from the University of Toronto in 1991, and the M.S. and Ph.D. degrees in Electrical Engineering from the California Institute of Technology in 1992 and 1997. He was an assistant, and then an associate, professor at Cornell University from 1997 to 2007. He is currently a full professor of automatic control at ETH Zurich. He is also a founding member and engineering fellow of systems architecture & algorithms at Kiva Systems. A creator of dynamic sculpture, his work has appeared at various international venues, including the National Gallery of Canada, the Venice Biennale, the Luminato Festival, Ars Electronica, and ideaCity.