Adaptive Open-Loop Aerobatic Maneuvers for Quadrocopters

Sergei Lupashin * Raffaello D’Andrea *

* Institute for Dynamic Systems and Control (IDSC), ETH Zurich, Zurich, Switzerland (e-mail: {sergeil, rdandrea}@ethz.ch)

Abstract: We describe a process for enabling quadrocopters to perform and improve upon aerobatic maneuvers. We describe such maneuvers as a set of desired keyframes and a parametrized input trajectory. The full state trajectory of the vehicle is left unspecified - only predefined partial-state keyframes are used to measure errors and to refine the primitive. A first-principles model is used to find nominal trajectory parameter values and a first-order correction matrix. We apply this method to extending previous work on vertical-plane 2D adaptive flips to a fully 3D adaptive maneuver. We also show how this method can be applied to finding trajectories for flips with matching non-zero initial and final velocities. Preliminary results are presented from simulation and from quadrocopters in the ETH Flying Machine Arena.

Keywords: Flying robots; Adaptive control; Model approximation; Learning control.

1. INTRODUCTION

Our goal is to develop a robust, straightforward process for enabling quadrocopters to perform and improve upon high-performance aerobatic maneuvers.

Recent research has been pushing the boundaries of aggressive quadrocopter control. Recent results include learning aggressive motions using iterative learning control methods described by Purwin and D’Andrea (2009), adaptive multi-flips by Lupashin et al. (2010), aggressive parameterized motions described by Mellinger et al. (2010), outdoor quadrocopter flips by Gillula et al. (2009), and dynamic inverted pendulum balancing and rhythmic primitives by Hehn and D’Andrea (2011) and by Schöllig et al. (2010), respectively. Autonomous helicopters have had a somewhat longer history, with several successful projects demonstrating aggressive helicopter aerobatics such as by Gerg (2008) and, with trajectories extracted from multiple demonstrations, by Abbeel et al. (2010) and by Tang et al. (2010). There are also successful projects for aggressive fixed- and flapping-wing vehicle control, for example by How et al. (2008a) and by Roberts et al. (2010).

Most of the above approaches require a feasible maneuver trajectory to be supplied, either via demonstration or from modeling. The objective of the learning algorithm is typically to make the vehicle follow a specified trajectory throughout the entire maneuver as well as possible. In contrast, we aim to improve the maneuvers by using a handful of discrete state observations at certain pre-specified key moments – keyframes. The feasibility of such an approach was demonstrated for adaptive planar flips in Lupashin et al. (2010).

For many maneuvers only a few keyframes, combined with a weak time optimality requirement, fully describe the desired motion: for example, for a flip we care only that the quadrocopter begins and ends in the same state plus a rotation – its motion during the flip itself is left unspecified. This allows us to use simplified models to construct maneuver input trajectories and learning strategies and reducing the required level of modeling of the specific mechanics of fast aerial vehicle motions. In this paper we focus only on initial and final state keyframes due to space constraints. We will discuss using intermediate and partial-state keyframes (e.g. vehicle must be at certain attitude but in any location) in future work.

The use of 3D pose keyframes is an extension of our previous work on adaptive flips described in Lupashin et al. (2010) where we used a planar 2D model combined...
with state errors described only within that plane. We generalize the 2D adaptive multi-flip maneuver so that it can be learned in all of the degrees of freedom and show how the same methodology can be applied to other aggressive motions.

A top-level overview of the described method is shown in Fig. 2. A user first describes a desired maneuver in terms of a set of keyframes and an input trajectory formulation. This leads to a set of fixed parameters and states and to a set of parameters to be learned. A first-order, first-principles model is used to find a nominal parameter set, and to calculate a correction matrix. Real-world iterative experiments then refine the nominal parameter set using the correction matrix in an attempt to drive real-world keyframe errors to zero.

We begin by introducing the generic process of keyframe, parameter input function, and error specification and by describing the parameter adaptation algorithm. We then introduce a simplified model for the 3D dynamics of a quadrocopter and the onboard controller. We apply the approach to two example motions: 3D adaptive hover-to-hover multi-flips (an extension of previous work) and an alternative flip formulation where the initial and final velocities match but are non-zero. We then show experimental results for running the algorithm in simulation and in the real world. We conclude with a discussion of some of the difficulties of the method presented and a future outlook.

2. KEYFRAMES AND ERROR CORRECTION

We start by defining a vector $\mathbf{c}$ that consists of all of the constant parameters of a maneuver. These are various design constants such as the maximum desired rate of rotation during a flip, the desired number of flips, etc. We define the vector $\mathbf{p}$ as the vector of parameters to be adjusted to improve the maneuver. These may include durations, angular accelerations, collective thrust values, initial/final velocities, or other values.

The inputs applied during the maneuver are then defined in terms of the maneuver constants $\mathbf{c}$ and adjustable parameters $\mathbf{p}$; for example, for quadrocopter inputs in the form of angular and collective accelerations as described later in this work, this input function takes the form of $(\ddot{a}_p(t), \ddot{a}_q(t), \ddot{a}_r(t), \tau_{col}(t)) = \mathbf{U}(\mathbf{c}, \mathbf{p}, t)$.

We want the vehicle to pass through certain states at specific moments. We formalize this as specifying a set of $N+1$ keyframes $\mathbf{X}_0..N$, with each keyframe defined as $\mathbf{X}_n(\mathbf{c}, \mathbf{p}) = (\mathbf{x}_n, t_n)$ where $\mathbf{x}_n$ is a desired state at time $t_n$.

The keyframe $\mathbf{X}_0$ is considered a special case: in this paper the vehicle is assumed to start exactly at the initial state described by $\mathbf{X}_0$. In other words, we assume that $\mathbf{x}(0) = \mathbf{x}_0$ and $t_0 = 0$. Given that the initial state and the input function $\mathbf{U}(\mathbf{c}, \mathbf{p}, t)$ are known, we can express the resulting state of the vehicle during an open-loop maneuver at time $t$ as $\mathbf{x}(t) = \mathbf{X}(\mathbf{c}, \mathbf{p}, t)$.

We can then collect all of the keyframe errors:

$$ E(\mathbf{c}, \mathbf{p}) = \begin{bmatrix} \mathbf{X}_1 - \mathbf{X}(\mathbf{c}, \mathbf{p}, t_1) \\ \vdots \\ \mathbf{X}_N - \mathbf{X}(\mathbf{c}, \mathbf{p}, t_N) \end{bmatrix} $$

Our objective is to find a parameter set $\mathbf{p}^*$ such that $E(\mathbf{c}, \mathbf{p}^*) = 0$. $\mathbf{p}^*$ is not known directly, since the actual system dynamics aren’t known, but we attempt to converge on it from a nearby parameter set $\mathbf{p}^0$. We note that, to first order,

$$ E(\mathbf{c}, \mathbf{p}) \approx E(\mathbf{c}, \mathbf{p}^*) + \frac{\partial E}{\partial \mathbf{p}}(\mathbf{p} - \mathbf{p}^*) \approx \frac{\partial E}{\partial \mathbf{p}}(\mathbf{p} - \mathbf{p}^*) $$

For conciseness, we define the nominal Jacobian at the nominal parameter set $\mathbf{p}^0$ as $\mathbf{J}$:

$$ J = \frac{\partial E}{\partial \mathbf{p}}|_{\mathbf{p}^0} $$

In practice, $\mathbf{J}$ can be calculated along with $\mathbf{p}^0$ in a generic manner using the nominal model of the dynamics, a numerical solver and small perturbations.

If $\mathbf{J}$ is invertible or if a suitable pseudo-inverse is used then to first order,

$$ \mathbf{p}^* \approx \mathbf{p} - \mathbf{J}^{-1}E(\mathbf{c}, \mathbf{p}) $$

motivating our correction strategy:

$$ \mathbf{p}^{i+1} = \mathbf{p}^i - \gamma J^{-1}E(\mathbf{c}, \mathbf{p}^i) $$

where $\gamma$ is a step size, $0 < \gamma \leq 1$, allowing for robustness to noise, non-systematic error and higher-order effects. Note that this is close to the policy gradient method (see, for example, Peters and Schaal (2006)), but where the gradient is calculated from a reference model rather than from measurements.

We can check the first-order convergence of this method: Let $\mathbf{e}^i = \mathbf{p}^i - \mathbf{p}^*$. Then, to first order,

$$ e^{i+1} = p^{i+1} - p^* $$

$$ = p^i - \gamma J^{-1}E(\mathbf{c}, \mathbf{p}^i + \mathbf{p}^* - \mathbf{p}^*) - p^* $$

$$ \approx e^i - \gamma J^{-1}(E(\mathbf{c}, \mathbf{p}^*) + Je^i) $$

$$ \approx (1 - \gamma)e^i $$

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

In this work, the state of each vehicle is defined as follows:
x = (x, y, z, \dot{x}, \dot{y}, \dot{z}, \phi, \theta, \psi) \quad (10)

where \( \phi, \theta \) and \( \psi \) are Z-Y-X Euler angles as described below.

### 3. QUADROCOPTER MODEL

While this method is not constrained to quadrocopters, we use this family of vehicles as our platform of choice for their agility, maintainability, mechanical simplicity, and robustness. We describe a modular model of quadrocopter dynamics where most levels of the model can be simplified resulting in reduced and more tractable sets of equations.

The simplification options given in the model allow for insight into the sensitivity of the correction matrix \( J \) to modeling errors. In addition, it allows us to test our correction strategy against known non-idealities. For example, we calculated correction matrices for hover-to-hover adaptive flips (Section 4) using both a model that included motor saturation and dynamics and one that excluded them. The difference between the two matrices and convergence rates was minimal—a good indicator that the correction strategy would be robust in real life.

We use a first-principles continuous-time model that approximates both the dynamics of the quadrocopter and the onboard controllers. Since the model is continuous with respect to both time and parameters it allows us to calculate gradients and use generic numerical optimization methods to find nominal aerobatic parameter values.

The coordinate system and rotor numbering convention used in the rest of this work is depicted in Fig. 3.

#### 3.1 Inputs

The model is based on the physical quadrocopters used in our lab. Each quadrocopter accepts four inputs: three angular body rates and a collective thrust. As discussed in Section 4, we focus on bang-bang type trajectories in acceleration inputs: angular acceleration commands \((\hat{\alpha}_p, \hat{\alpha}_q, \hat{\alpha}_r)\) and a collective thrust command \(\hat{\tau}_{coll}\). The angular acceleration commands are integrated up from an initial value of 0, resulting in a set of angular rate commands \((\dot{\hat{p}}, \dot{\hat{q}}, \dot{\hat{r}})\) to be followed by the onboard controller.

#### 3.2 Angular Rate Control

Onboard rate gyros provide sensor data for feedback on angle rate, helping to mitigate unmodeled or unpredictable aerodynamic effects, motor dynamics, etc. Given the commanded \((\dot{\hat{p}}, \dot{\hat{q}}, \dot{\hat{r}})\) and current \((p, q, r)\) body rates, the desired onboard angular accelerations \((\hat{\alpha}_p, \hat{\alpha}_q, \hat{\alpha}_r)\) are

\[
\begin{align*}
\hat{\alpha}_p &= P_p (\dot{\hat{p}} - p) + I_p \int (\dot{\hat{p}} - p) dt \\
\hat{\alpha}_q &= P_q (\dot{\hat{q}} - q) + I_q \int (\dot{\hat{q}} - q) dt \\
\hat{\alpha}_r &= P_r (\dot{\hat{r}} - r)
\end{align*}
\]

where \( P_p, q, r \) are proportional gains and \( I_p, q, r \) are integral gains. We may choose to simplify the model by bypassing the angular rate feedback loop; in this case, set \((\hat{\alpha}_p, \hat{\alpha}_q, \hat{\alpha}_r) = (\hat{\alpha}_p, \hat{\alpha}_q, \hat{\alpha}_r)\).

#### 3.3 Accelerations

From first principles, the propeller forces \(\tau_{1-4}\) induce the following angular accelerations:

\[
\dot{\omega} = \begin{bmatrix} \dot{\hat{p}} \\ \dot{\hat{q}} \\ \dot{\hat{r}} \end{bmatrix} = I^{-1} \begin{bmatrix} l(\tau_2 - \tau_4) \\ l(\tau_3 - \tau_1) \\ \kappa(\tau_1 - \tau_2 + \tau_3 - \tau_4) \end{bmatrix} - I^{-1} \omega \times \mathbf{I} \quad (14)
\]

where \( I \) is the inertia matrix of the vehicle (typically diagonal), \( l \) is the vehicle center to rotor distance and \( \kappa \) is an experimentally determined constant. The propeller forces also result in the acceleration

\[
\ddot{z}_b = (\tau_1 + \tau_2 + \tau_3 + \tau_4)/m \quad (15)
\]

where \( m \) is the mass of the vehicle.

To convert desired onboard angular accelerations \((\hat{\alpha}_p, \hat{\alpha}_q, \hat{\alpha}_r)\) and a commanded \(\hat{\tau}_{coll}\) to motor thrust commands, we invert (14) and (15):

\[
\begin{align*}
\hat{\tau}_1 &= (\hat{\tau}_{coll} + \hat{\mu}_r)/\kappa - 2\hat{\mu}_q/l/4 \\
\hat{\tau}_2 &= (\hat{\tau}_{coll} - \hat{\mu}_r)/\kappa + 2\hat{\mu}_p/l/4 \\
\hat{\tau}_3 &= (\hat{\tau}_{coll} + \hat{\mu}_r)/\kappa + 2\hat{\mu}_q/l/4 \\
\hat{\tau}_4 &= (\hat{\tau}_{coll} - \hat{\mu}_r)/\kappa - 2\hat{\mu}_p/l/4
\end{align*}
\]

where \( \hat{\mu}_p, \hat{\mu}_q, \hat{\mu}_r \) are desired moments that are calculated from desired onboard angular accelerations \((\hat{\alpha}_p, \hat{\alpha}_q, \hat{\alpha}_r)\) and the current body rates \(\omega\):

\[
\begin{bmatrix} \hat{\mu}_p \\ \hat{\mu}_q \\ \hat{\mu}_r \end{bmatrix} = I \begin{bmatrix} \hat{\alpha}_p \\ \hat{\alpha}_q \\ \hat{\alpha}_r \end{bmatrix} + I^{-1} \omega \times \mathbf{I} \quad (20)
\]

The \( \omega \times \mathbf{I} \) cross term is typically minor and can be ignored if \( I \) is sufficiently close to a scaled identity matrix. Simplifying out this effect also results in \( \dot{p}, \dot{q}, \dot{r} \) being coupled only via potential motor saturation constraints and/or motor dynamics. We found that including this term in our model has little effect on the gradient matrix.

#### 3.4 Motors

Each rotor produces a thrust \(\tau_{min} \leq \tau_{1-4} \leq \tau_{max}\). For simplicity, we assume identical rotors. In addition to a thrust force, each rotor also produces a drag-induced moment linear to the thrust force \(\kappa \tau_i\) as mentioned in (14).
Our vehicles use brushless motors and motor speed motor controllers that can be treated as a first order system:

$$\dot{\tau}_i = P_r (\tau_i - \tau_i)$$

(21)

where \(P_r\) is an experimentally measured constant.

We’ve found that motors are faster to speed up than to slow down (most motor controllers do not do active braking). To model this, we can define a \(P_{r,up} \geq P_{r,down}\).

To bypass motor dynamics, we may set \(\tau_{\text{up}} = \tau_{\text{down}}\). To ignore motor saturation, we may set \(\tau_{\text{max}} = -\tau_{\text{min}} = \infty\).

3.5 Translation

Translational motion in the global frame is given by:

$$\begin{bmatrix} \dot{x}_g \\ \dot{y}_g \\ \dot{z}_g \end{bmatrix} = \Omega^g R \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$$

(22)

where \(\Omega^g R\) is a rotation matrix as follows.

3.6 Attitude

Attitude is represented in the form of Z-Y-X Euler angles \(\phi, \theta, \psi\). The rotation matrix is defined as

$$\Omega^g R = \begin{bmatrix} c_\nu c_\psi & s_\nu s_\psi & -c_\psi \\ s_\psi c_\nu & c_\psi s_\psi & s_\nu \\ -s_\nu c_\theta & -s_\nu s_\theta & c_\theta \end{bmatrix}$$

(23)

where \(c_\nu\) and \(s_\nu\) are shorthand for \(\cos \nu\) and \(\sin \nu\), respectively. The Euler rates are defined as (Diebel (2006)).

$$\begin{bmatrix} \dot{c}_\psi \\ \dot{s}_\psi \\ \dot{c}_\theta \\ \dot{s}_\theta \end{bmatrix} = \frac{1}{c_\theta} \begin{bmatrix} c_\psi & s_\psi & 0 \\ -c_\psi s_\theta c_\nu - s_\psi c_\nu c_\theta & c_\psi s_\theta s_\nu - s_\psi c_\nu c_\theta & c_\psi c_\nu s_\theta & c_\psi c_\theta \\ -c_\psi s_\theta s_\nu - s_\psi c_\nu c_\theta & c_\psi s_\theta c_\nu - s_\psi c_\nu c_\theta & c_\psi s_\nu s_\theta & c_\psi c_\theta \\ s_\psi & c_\psi & 0 \end{bmatrix} \begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix}$$

(24)

Note that this representation suffers from a singularity when \(c_\theta = 0\). This can be avoided by using a different formulation of attitude such as a normalized quaternion, not used in the experiments described in this work.

4. HOVER-TO-HOVER FLIPS

We apply our method to creating an adaptive hover-to-hover flip maneuver, stabilized in all degrees of motion. We want the quadrocopter to start from rest (static hover), rotate a number of times about it’s \(x_h\) axis, and return to the starting point with the same final state as at the beginning. This is an extension of our previous work (Lupashin et al. (2010)) that demonstrated how such a maneuver can be formulated and adapted as a 2D vertical-plane motion.

There are several possible levels at which input trajectories can be defined: as motor commands, as accelerations (angular and collective), as velocities (angular or collective), or as even higher-level commands. A higher level of commands provides more opportunity for closed-loop control while lower command levels have exponentially faster time responses. We choose the acceleration control level, as this allows for good balance between allowing for feedback via the rate gyros and for relatively fast time responses.

Drawing inspiration from bang-bang control, we focus on trajectories where the inputs are always saturated. The thrust constraints on the motors result in coupling between the saturation limits of the inputs, leading to coupled control envelopes as shown in Fig. 4. For bang-bang type input trajectories, the commands should always lie on the edge of the allowable control envelope, with the exception of when saturation is reached at a higher control level. We relax this requirement for controlling secondary (lateral) degrees of freedom, as using saturated commands simultaneously for all of the inputs can easily lead to secondary effects such as motor response becoming dominant. In addition, since the vehicle itself uses feedback control to follow the commanded inputs, we shrink the control envelope by a predefined value in order to reserve some control effort for feedback and modeling errors. For the rest of this paper we refer to the maximum and minimum collective accelerations possible within the reduced control envelope as \(\vec{\beta}\) and \(\vec{\beta}\), respectively, where \(4\tau_{\text{min}} \leq m\vec{\beta} < m\vec{\beta} \leq 4\tau_{\text{max}}\) and \(m\) is the mass of the vehicle.

In practice, we represent an input function for a given maneuver as a list of switch events. Each entry describes the switch of one or more of the inputs to a new value at a certain time. For the primary degrees of freedom, the inputs are switched to always lie on the edge of the reduced control envelope. For secondary degrees of freedom, the inputs are switched between some relatively small adjustable values.

For this maneuver the external parameter set \(c\) includes the maximum and minimum usable collective thrust \(\vec{\beta}\) and \(\vec{\beta}\) physical parameters of the quadrocopter such as the center-motor distance \(l\) and mass \(m\), the maximum desired rotation rate to be reached during the flip \(c_{p_{\max}}\) and the desired number of flips \(c_{R}\).

The keyframes are then

$$\mathcal{K}_0(c, \mathbf{p}) = (0, 0)$$

(25)

$$\mathcal{K}_1(c, \mathbf{p}) = (0, 0, 0, 0, 0, 0, 2\pi c_{R}, 0, 0, t_{\text{end}})$$

(26)

Extending the previous work, we define the input function in five stages (see Fig. 5). The roll axis commands are...
The remaining inputs are then defined as follows:

\[
\hat{a}_\phi(t) = \begin{cases} 
\text{accelerate} & F_1(p_5, 0, p_1) \\
\text{start rotate} & F_1(p_6, p_1, T_2) \\
\text{stop rotate} & F_1(p_7, p_1 + T_2 + p_2, T_4) \\
\text{otherwise} & 0
\end{cases}
\]

(31)

\[
\hat{a}_\psi(t) = \begin{cases} 
\text{accelerate} & F_2(p_8, 0, p_1 + p_4, p_1) \\
\text{recover} & F_2(p_8, t_{end} - s_p - p_1 + p_4, p_1 + t_{end}) \\
\text{otherwise} & 0
\end{cases}
\]

(32)

where

\[
T_2 = (c_{pmax} - p_1\hat{a}_{p,accel})/\hat{a}_{p,startrot} \\
T_4 = (c_{pmax} + p_4\hat{a}_{p,stoprot})/\hat{a}_{p,stoprot} \\
t_{end} = p_1 + T_2 + p_2 + T_4 + p_4
\]

(33)

(34)

(35)

5. ALTERNATIVE FLIP FORMULATION

We can also apply our method to finding a nominal “continuous flip”: a maneuver where the final state is not necessarily static hover but instead the initial state plus a \(2\pi\) rotation in roll.

The simplest form of a continuous flip is a constant-rotation form, controlled by turning the thrust on and off at the right phase offset with respect to the periodic maneuver. Algebraic analysis with a first-principles model reveals that to be able to do this \((\tau_{max} - \tau_{min}) \geq \pi gm/4\), independent of the rate of rotation at which the flip is performed. Unfortunately, this is beyond the current capabilities of the quadrocopters in our lab, which have a nominal \(\tau_{max}\) of around 2.1 gm/4 before dynamic losses.

We use the next simplest form: the rate of rotation should be 0 when the vehicle is at the beginning of each period and should be the fastest in the middle. The intuition here is that we want to spend more time pointed up (such that the vehicle has more time to accelerate upward against gravity) and less time pointed down. Since we don’t know the velocities at the beginning of each continuous flip, but want the initial and final conditions to be exactly the same, the keyframes for this maneuver are

\[
K_0(c, p) = (0, 0, 0, 0, p_2, p_3, 0, 0, 0, 0)
\]

(36)

\[
K_1(c, p) = (0, 0, 0, 0, p_2, p_3, 2\pi, 0, 0, t_{end})
\]

(37)

which describe the constraint that some of the initial and final velocities are unknown but must match at the beginning and at the end of the maneuver:

Fig. 6. Different simulated continuous flip trajectories possible from changing the collective/angular acceleration trade-off parameter \(c_A\). The values of \(c_A\) used for each trajectory, from smallest loop to largest, are 0.2, 0.13 and 0.08. The vehicles here are depicted as 0.34 m across, shown every 20 ms of the maneuver.
\[
\hat{y}_g(0) = \hat{y}_g(t_{\text{end}}) = p_2 \\
\hat{z}_g(0) = \hat{z}_g(t_{\text{end}}) = p_3
\]

An external parameter \( c_A \), \( 0 < c_A < 1 \) provides control over how thrust should be allocated between collective and rotational accelerations. A value of 0 for \( c_A \) means that all thrust effort should be used for collective acceleration, resulting in an impossible "infinite" flip, since there is no control effort remaining for angular acceleration. A higher value of \( c_A \) means a faster flip, but may result in angular rate saturations and is also more sensitive to timing errors. Fig. 6 shows some possible different continuous flip trajectories when all other external parameters are held constant.

The collective thrust and roll acceleration input functions are as follows:

\[
\hat{\tau}_{\text{col}}(t) = \begin{cases} 
\text{coast} & m(\beta + (\beta - \tilde{\beta})c_A/2) \\
\text{otherwise} & m(\beta - (\beta - \tilde{\beta})c_A/2)
\end{cases}
\]

\[
\hat{\alpha}_p(t) = p_4 \text{sgn}(t - t_{\text{end}}/2)(\beta - \tilde{\beta})c_A ml/4 I_{xx}
\]

where the duration of the maneuver is

\[
t_{\text{end}} = \sqrt{\frac{32\pi I_{xx}}{(\beta - \tilde{\beta})c_A ml}}
\]

The lateral degrees of freedom are stabilized similarly to the adaptive flip:

\[
\hat{\alpha}_q(t) = \begin{cases} 
\text{accelerate} & F_1(p_5, 0, p_6) \\
\text{coast} & F_1(p_6, p_0, p_1 - p_0) \\
\text{recover} & F_1(p_7, p_1, t_{\text{end}} - p_1)
\end{cases}
\]

\[
\hat{\alpha}_r(t) = F_1(p_8, 0, t_{\text{end}})
\]

The complete input function is depicted in Fig. 7.

6. EXPERIMENTS

The algorithm was tested in the ETH Flying Machine Arena, a 10x10x10 m testbed for aerial robotics. The space

![Image](Image)

Fig. 7. Continuous adaptive flip profile. The parameters \( p_{2,3} \) specify initial/final \( \hat{y}_g \) and \( \hat{z}_g \), respectively.

Fig. 8. Evolution of errors for a full-3D adaptive triple flip executed on a real quadrocopter at the ETH Flying Machine Arena.

is organized similar to How et al. (2008b), with a motion capture system providing high-framerate, high-accuracy position and attitude information about every vehicle. See Lupashin et al. (2010) for more details about the testing environment.

For each maneuver, off-board code generates a list of switch events and maneuver descriptors (duration, initial inputs). These are transmitted to the onboard controller. A special command then triggers the execution of the maneuver. Command and motion capture system latencies are well-known, allowing for accurate observation of vehicle state at any point during the maneuver. A symmetric low-pass filter is used to filter the measurements and provide more accurate velocity estimates. For the parameter adaptation algorithm, we take only the filtered state measurements at specific moments of interest: at the beginning \( (t = 0) \) and at the end \( (t = t_{\text{end}}) \) of the maneuver.

At each iteration, we apply the adaptation method described by (5). If any of the parameters in the new parameter vector are outside of their constraints, the step size \( \gamma \) is scaled down appropriately and the correction is recalculated. A consistent saturation of parameters usually indicates that the desired maneuver is not practically feasible.

An experimental run of a hover-to-hover adaptive flip maneuver is shown in Fig. 8. The step size was defined as follows: \( \gamma = \max(.05, 1/N) \) where \( N \) is the number of the iteration (first iteration being 1). The algorithm converges on a good solution within several iterations.

7. DISCUSSION

Although in principle the method presented here is quite generic and robust to modeling errors, creating a parametrized input trajectory formulation that results in a robust learning strategy remains a difficult and tedious task.

In particular, it is important to keep parameters from significantly affecting each other. This is usually straight-
forward to achieve, for example by normalizing parameters with respect to the time durations during which they have effect, as done in Section 4.

8. CONCLUSION

This paper described the outlines of a process for specifying and adapting aerobatic maneuvers from sparse keyframes. While there are many improvements possible, the methodology described above attempts to use a different perspective for approaching complex control problems where precise modeling is either too tedious or impractical. The approach has good robustness against modeling errors but leaves the user a choice between a range of models of various levels of sophistication.

We hope that this approach can complement existing methods for performing and improving upon difficult motions, such as the work done on learning aerobatics from demonstrations or existing iterative learning control-inspired approaches. In the future, we hope to address some of the shortcomings of the described approach and to demonstrate it on more complex aerobatic maneuvers.

ACKNOWLEDGEMENTS

We would like to thank Markus Hehn and Angela Schöllig for their contributions to the ETH Flying Machine Arena testbed and for their comments.

REFERENCES


