A computationally efficient motion primitive for quadrocopter trajectory generation

Mark W. Mueller, Markus Hehn, and Raffaello D'Andrea

***Abstract*—A method is presented for the rapid generation and feasibility verification of motion primitives for quadrocopters and similar multirotor vehicles. The motion primitives are defined by the quadrocopter's initial state, the desired motion duration, and any combination of components of the quadrocopter's position, velocity and acceleration at the motion's end. Closed form solutions for the primitives are given, which minimize a cost function related to input aggressiveness. Computationally efficient tests are presented to allow for rapid feasibility verification. Conditions are given under which the existence of feasible primitives can be guaranteed a priori. The algorithm may be incorporated in a high-level trajectory generator, which can then rapidly search over a large number of motion primitives which would achieve some given high-level goal. It is shown that a million motion primitives may be evaluated and compared per second on a standard laptop computer. The motion primitive generation algorithm is experimentally demonstrated by tasking a quadrocopter with an attached net to catch a thrown ball, evaluating thousands of different possible motions to catch the ball.**

I. Introduction

QUADROCOPTERS offer exceptional agility, with typically high thrust-to-weight ratios, and large potential for angular acceleration due to the outward mounting of the propellers. This allows them to perform complex and highly dynamic tasks, for example aerial manipulation [1] and cooperative aerial acrobatics [2]. The ability to hover, and the safety offered by small rotors storing relatively little energy [3], make quadrocopters attractive platforms for aerial robotic tasks that involve the navigation of tight, cluttered environments (for example, [4], [5]).

A key feature required for the use of these vehicles under complex conditions is a trajectory generator. The trajectory generator is tasked with computing flight paths that achieve the task objective, while respecting the quadrocopter dynamics. The trajectory must also be collision-free, and there could be additional requirements imposed on the motion by the sensing modalities (for example, limits on the quadrocopter velocity imposed by an onboard camera). This trajectory planning problem is complicated by the underactuated and nonlinear nature of the quadrocopter dynamics, as well as potentially complex task constraints that may consist of variable manoeuvre durations, partially constrained final states, and non-convex state constraints. In addition, dynamic environments or large disturbances may require the re-computation or adaptation of trajectories in real time, thus limiting the time available for computation.

The authors are with the Institute for Dynamic Systems and Control, ETH Zurich, Sonneggstrasse 3, 8092 Zurich, Switzerland.
{mwm,hehnm,rdandrea}@ethz.ch
(c) 2015 IEEE

Active research in this field has yielded numerous trajectory generation algorithms, focusing on different trade-offs between computational complexity, the agility of the possible motions, the level of detail in which manoeuvre constraints can be specified, and the ability to handle complex environments.

Broadly speaking, a first group of algorithms handles the trajectory generation problem by decoupling geometric and temporal planning: in a first step, a geometric trajectory without time information is constructed, for example using lines [6], polynomials [7], or splines [8]. In a second step, the geometric trajectory is parametrised in time in order to guarantee feasibility with respect to the dynamics of quadrocopters.

A second group of algorithms exploits the differential flatness of the quadrocopter dynamics in order to derive constraints on the trajectory, and then solves an optimization problem over a class of trajectories, for example minimum snap [9], minimum time [10], shortest path under uncertain conditions [11], or combinations of position derivatives [5]. In [12] a search over parameters is proposed for quadrocopter motion planning, including trajectories where the position is described by polynomials in time. A dynamic inversion based controller is proposed in [13] to directly control a quadrocopter's position and orientation, and this controller is exploited in [14]. For a broader discussion of differential flatness see e.g. [15], [16], and e.g. [17], [18] for generic trajectory generation methods for differentially flat systems.

A common property of these methods is that they impose a rigid structure on the end state, for example fixing the final state and allowing a fixed or varying manoeuvre duration, or by specifying the goal state with convex inequalities. Many quadrocopter applications do not, however, impose such structured constraints; instead the set of states that achieve the

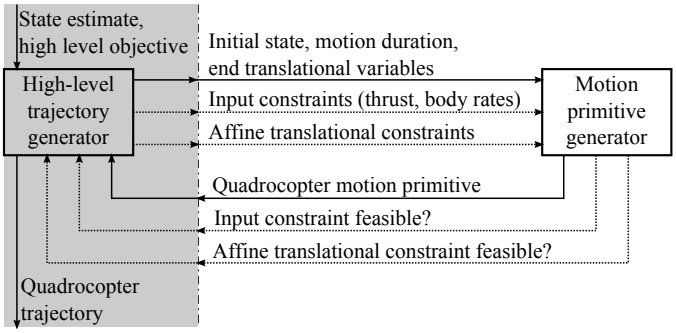


Fig. 1. The presented algorithm aims to provide computationally inexpensive motion primitives, which may then be incorporated by a high-level trajectory generator. The focus of this paper is on the right-hand-side (unshaded) part of this diagram.

application might be non-convex, or even disjoint. Furthermore, the conditions on the final state necessary to achieve a task may be time-dependent (for example when the task objective involves interaction with a dynamic environment). Methods relying on convex optimisation furthermore require the construction of (conservative) convex approximations of constraints, potentially significantly reducing the space of feasible trajectories.

This paper attempts a different approach to multicopter trajectory generation, where the constraints are not explicitly encoded at the planning stage. Instead, the focus is on developing a computationally light-weight and easy-to-implement motion primitive, which can be easily tested for constraints violation, and allows significant flexibility in defining initial and final conditions for the quadrocopter. The low computational cost may then be exploited by searching over a multitude of primitives to achieve some goal. Each primitive is characterised by the quadrocopter's initial state, a duration, and a set of constraints on the quadrocopter's position, velocity, and/or acceleration at the end of the primitive.

The approach is illustrated in Fig. 1. The high-level trajectory generator is tasked with evaluating motion primitives, and specifically with defining the constraints on the motion primitives to solve the given high-level goal. The high-level trajectory generator must also encode the behaviour for dealing with infeasible motion primitives. As an example, the high-level trajectory generator may generate a large number of motion primitives, with varying durations and end variables, to increase the probability of finding a feasible motion primitive. These motion primitives are generated in a two-step approach: First, a state-to-state planner is used to generate a motion while disregarding feasibility constraints. In the second step, this trajectory is checked for feasibility. The first step is solved for in closed form, while a computationally efficient recursive test is designed for the feasibility tests of the second step.

The state-to-state motion primitives generated in the first step are closely related to other algorithms exploiting the differential flatness of the quadrocopter dynamics to plan position trajectories that are polynomials in time (e.g. [5], [9], [12]). In this paper an optimal control problem is solved, whose objective function is related to minimizing an upper bound of the product of the quadrocopter inputs, to yield position trajectories characterised by fifth order polynomials in time. A key property that is then exploited is that the specific polynomials allow for the rapid verification of constraints on the system's inputs, and constraints on the position, velocity, and/or acceleration.

The benefits of this approach are twofold: first, a unified framework is given to generate trajectories for arbitrary manoeuvre duration and end state constraints, resulting in an algorithm which can be easily implemented across a large range of trajectory generation problems. Secondly, the computational cost of the approach is shown to be very low, such that on the order of one million motion primitives per second can be generated and tested for feasibility on a laptop computer with an unoptimised implementation.

The algorithm therefore lends itself to problems with significant freedom in the end state. In this situation, the designer can apply the presented approach to rapidly search over the space of end states and trajectory durations which would achieve the high level goal. This ability to quickly evaluate a vast number of candidate trajectories is achieved at the expense of not directly considering the feasibility constraints in the trajectory generation phase, but rather verifying feasibility a posteriori.

For certain classes of trajectories, explicit guarantees can be given on the existence of feasible motion primitives as a function of the problem data. Specifically, for rest-to-rest manoeuvres, a bound on the motion primitive duration is explicitly calculated as a function of the distance to be translated and the system's input constraints. Furthermore, bounds on the velocity during the rest-to-rest manoeuvre are explicitly calculated, and it is shown that the position feasibility of rest-to-rest trajectories can be directly asserted if the allowable flight space is convex.

An experimental demonstration of the algorithm is presented where the motion primitive generator is encapsulated in a high-level trajectory generation algorithm. The goal is for a quadrocopter with an attached net to catch a thrown ball. The catching trajectories must be generated in real time, because the ball's flight is not repeatable. Furthermore, for a given ball trajectory, the ball can be caught in many different ways (quadrocopter positions and orientations). The computational efficiency of the presented approach is exploited to do an exhaustive search over these possibilities in real time.

An implementation of the algorithm presented in this paper in both C++ and Python is made available at [19].

This paper follows on previous work presented at conferences [20], [21]. A related cost function, the same dynamics model, and a related decoupled planning approach were presented in [20]. Preliminary results of the fundamental state-to-state motion primitive generation algorithm were presented in [21]. This paper extends these previous results by presenting

- conditions under which primitives are guaranteed to be feasible;
- an investigation into the completeness of the approach, by comparing rest-to-rest trajectories to the time optimal rest-to-rest trajectories; and
- presenting a challenging novel demonstration to show the capabilities of the approach.

The remainder of this paper is organised as follows: the quadrocopter model and problem statement are given in Section II, with the motion primitive generation scheme presented in Section III. A computationally efficient algorithm to determine feasibility of generated trajectories is presented in Section IV. The choice of coordinate system is discussed in Section V. In Section VI classes of problems are discussed where the existence of feasible trajectories can be guaranteed. The performance of the presented approach is compared to the system's physical limits in Section VII. The computational cost of the algorithm is measured in Section VIII, and the demonstration of catching a ball is presented in Section IX. A conclusion is given in Section X.

II. System dynamics and problem statement

This section describes the dynamic model used to describe the quadrocopter's motion, including constraints on the quadrocopter inputs. A formal statement of the problem to be solved in this paper is then given, followed by an overview of the solution approach.

A. Quadrocopter dynamic model

The quadrocopter is modelled as a rigid body with six degrees of freedom: linear translation along the orthogonal inertial axes, $\boldsymbol{x} = (x_1, x_2, x_3)$, and three degrees of freedom describing the rotation of the frame attached to the body with respect to the inertial frame, defined by the proper orthogonal matrix $\mathbf{R}$. Note that the notation $\boldsymbol{x} = (x_1, x_2, x_3)$ is used throughout this paper to compactly express the elements of a column vector.

The control inputs to the system are taken as the scalar total thrust produced f, for simplicity normalised by the vehicle mass and thus having units of acceleration; and the body rates expressed in the body-fixed frame as $\boldsymbol{\omega} = (\omega_1, \omega_2, \omega_3)$, as are illustrated in Fig. 2. It is assumed that high bandwidth controllers are used to track these angular rate commands. Then by separation of timescales, because of the vehicles' low rotational inertia and their ability to produce large torques, it is assumed that the angular rate commands are tracked perfectly and that angular dynamics may be neglected. The quadrocopter's state is thus nine dimensional, and consists of the position, velocity, and orientation.

Although more complex quadrocopter models exist that incorporate, for example, aerodynamic drag [22] or propeller speeds [23], the preceding model captures the most relevant dynamics, and yields tractable solutions to the trajectory generation problem. Furthermore, in many applications (for example, model predictive control) such a simple model is sufficient, with continuous replanning compensating for modelling inaccuracies.

The differential equations governing the flight of the quadrocopter are now taken as those of a rigid body [24]

$$\ddot{\boldsymbol{x}} = \mathbf{R}\, \boldsymbol{e}_3 f + \boldsymbol{g} \tag{1}$$

$$\dot{\mathbf{R}} = \mathbf{R}\, [\![\boldsymbol{\omega}\times]\!] \tag{2}$$

with $\boldsymbol{g}$ the acceleration due to gravity as expressed in the inertial coordinate frame, $\boldsymbol{e}_3 = (0, 0, 1)$ a constant vector in the body-fixed frame, as illustrated in Fig. 2. Finally, $[\![\boldsymbol{\omega}\times]\!]$ is the skew-symmetric matrix form of the vector cross product such that

$$[\![\boldsymbol{\omega}\times]\!] = \begin{bmatrix} 0 & -\omega_3 & \omega_2 \\ \omega_3 & 0 & -\omega_1 \\ -\omega_2 & \omega_1 & 0 \end{bmatrix}. \tag{3}$$

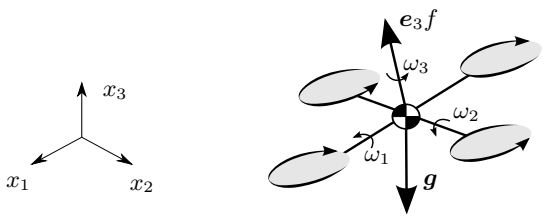


Fig. 2. Dynamic model of a quadrocopter, acted upon by gravity $\boldsymbol{g}$, a thrust force f pointing along the (body-fixed) axis $\boldsymbol{e}_3$; and rotating with angular rate $\boldsymbol{\omega} = (\omega_1, \omega_2, \omega_3)$, with its position in inertial space given as (x_1, x_2, x_3).

It should be noted that the preceding model may also be applied to other multirotor vehicles, such as hexa- and octocopters. This is because the high-bandwidth angular rate controller that maps angular velocity errors to motor forces effectively hides the number and locations of the propellers.

1) Feasible inputs: The achievable thrust f produced by the vehicle lies in the range

$$0 \leq f_{\min} \leq f \leq f_{\max} \tag{4}$$

where $f_{\min}$ is non-negative because of the fixed sense of rotation of the fixed-pitch propellers. The magnitude of the angular velocity command is taken to be constrained to lie within a ball:

$$\|\boldsymbol{\omega}\| \leq \omega_{\max} \tag{5}$$

with $\|\cdot\|$ the Euclidean norm. This limit could be due, for example, to saturation limits of the rate gyroscopes, or motion limits of cameras mounted on the vehicle. Alternatively, a designer may use this value as a tuning factor, to encode the dynamic regime where the low-order dynamics of (1)-(2) effectively describe the true quadrocopter. The Euclidean norm is chosen for computational convenience, specifically invariance under rotation.

B. Problem statement

Define $\sigma(t)$ to be translational variables of the quadrocopter, consisting of its position, velocity and acceleration, such that

$$\sigma(t) = (\boldsymbol{x}(t), \dot{\boldsymbol{x}}(t), \ddot{\boldsymbol{x}}(t)) \in \mathbb{R}^9. \tag{6}$$

Let T be the goal duration of the motion, and let $\hat{\sigma}_i$ be components of desired translational variables at the end of the motion, for some $i \in \mathcal{I} \subseteq \{1, 2, \ldots, 9\}$. Let the trajectory goal be achieved if

$$\sigma_i(T) = \hat{\sigma}_i \; \forall i \in \mathcal{I}. \tag{7}$$

Furthermore, the quadrocopter is subject to N_c translational constraints of the form

$$a_j^T \sigma(t) \leq b_j \text{ for } t \in [0, T]\,,\; j = 1, 2, \ldots, N_c. \tag{8}$$

An interpretation of these translational constraints is given at the end of this section.

The problem addressed in this paper is to find quadrocopter inputs $f(t)$, $\boldsymbol{\omega}(t)$ over $[0, T]$, for a quadrocopter starting at some initial state consisting of position, velocity, and orientation, at time $t = 0$ to an end state at time T satisfying (7), while satisfying throughout the trajectory:

- the vehicle dynamics (1)-(2),
- the input constraints (4)-(5), and
- the N_c affine translational constraints (8).

Discussion: It should be noted that the nine quadrocopter state variables as described in Section II-A (three each for position, velocity and orientation) are not the nine translational variables. Only eight components of the state are encoded in the translational variables, consisting of the quadrocopter's position, its velocity, and two components of the orientation (which are encoded through the acceleration). These two orientation components are those that determine the direction of the quadrocopter's thrust vector – given an acceleration value, the quadrocopter's thrust direction $\mathbf{R}\boldsymbol{e}_3$ can be recovered through (1). These are the same components encoded by the Euler roll and pitch angles. The translational variables do not encode the quadrocopter's rotation about the thrust axis (the Euler yaw angle).

These variables are chosen because they are computationally convenient, whilst still being useful to encode many realistic problems. Two example problems are given below:

(i) To-rest manoeuvre: let the goal be to arrive at some point in space, at rest, at time T. Then all nine translational variables will be specified in (7), with specifically the components corresponding to velocity and acceleration set to zero. Similarly, if the goal is simply to arrive at a point in space, but the final velocity and acceleration do not matter, only the first three components of $\hat{\sigma}$ are specified.

(ii) Landing on a moving platform: to land the quadrocopter on a moving, possibly slanted platform, the goal end position and velocity are set equal to those of the platform at time T. The end acceleration is specified to be such that the quadrocopter's thrust vector e_3 is parallel to the normal vector of the landing platform. Then the quadrocopter will arrive with zero relative speed at the platform, and touch down flatly with respect to the platform.

The affine translational constraints of (8) are also chosen for computational convenience. They allow to encode, for example, that the position may not intersect a plane (such as the ground, or a wall), or specify a box constraint on the vehicle velocity. Constraints on the acceleration, in conjunction with (4), can be interpreted as limiting the tilt of the quadrocopter, by (1).

III. Motion primitive generation

Given an end time T and goal translational variables $\hat{\sigma}_i$, the dynamic model of Section II-A is used to generate motion primitives guiding the quadrocopter from some initial state to a state achieving the goal translational variables. The input constraints, and the affine state constraints, are ignored at this stage, and the motion primitives are generated in the quadrocopter's jerk. Each of the three spatial axes is solved for independently. The generated motion primitive minimises a cost value for each axis independently of the other axes, but the total cost is shown to be representative of the aggressiveness of the true system inputs. Constraints on the input and state are considered in Sections IV and VI.

A. Formulating the dynamics in jerk

We follow [10] in planning the motion of the quadrocopter in terms of the jerk along each of the axes, allowing the system to be considered as a triple integrator in each axis. By ignoring the input constraints, the axes can be decoupled, and motions generated for each axis separately. These decoupled axes are then recombined in later sections when considering feasibility. This subsection will describe how to recover the thrust and body rates inputs from such a thrice differentiable trajectory.

Given a thrice differentiable motion $\boldsymbol{x}(t)$, the jerk is written as $\boldsymbol{j} = \dddot{\boldsymbol{x}} = (\dddot{x}_1, \dddot{x}_2, \dddot{x}_3)$. The input thrust f is computed by applying the Euclidean norm to (1):

$$f = \|\ddot{\boldsymbol{x}} - \boldsymbol{g}\|. \tag{9}$$

Taking the first derivative of (1) and (9), yields

$$\boldsymbol{j} = \mathbf{R}[\![\boldsymbol{\omega}\times]\!]\boldsymbol{e}_3 f + \mathbf{R}\boldsymbol{e}_3\dot{f} \tag{10}$$

$$\dot{f} = \boldsymbol{e}_3^T\mathbf{R}^{-1}\boldsymbol{j}. \tag{11}$$

After substitution, and evaluating the product $[\![\boldsymbol{\omega}\times]\!]\boldsymbol{e}_3$, it can be seen that the jerk $\boldsymbol{j}$ and thrust f fix two components of the body rates:

$$\begin{bmatrix}\omega_2\\-\omega_1\\0\end{bmatrix} = \frac{1}{f}\begin{bmatrix}1&0&0\\0&1&0\\0&0&0\end{bmatrix}\mathbf{R}^{-1}\boldsymbol{j}. \tag{12}$$

Note that ω_3 does not affect the linear motion, and is therefore not specified. Throughout the rest of the paper, it will be taken as $\omega_3 = 0$.

B. Cost function

The goal of the motion primitive generator is to compute a thrice differentiable trajectory which guides the quadrocopter from an initial state to a (possibly only partially defined) final state in a final time T, while minimizing the cost function

$$J_\Sigma = \frac{1}{T}\int_0^T \|\boldsymbol{j}(t)\|^2\, dt. \tag{13}$$

This cost function has an interpretation as an upper bound on the average of a product of the inputs to the (nonlinear, coupled) quadrocopter system: rewriting (12), and taking $\omega_3 = 0$ gives

$$\begin{aligned} f^2\|\boldsymbol{\omega}\|^2 &= \left\| f\begin{bmatrix}\omega_2\\-\omega_1\\0\end{bmatrix}\right\|^2 = \left\|\begin{bmatrix}1&0&0\\0&1&0\\0&0&0\end{bmatrix}\mathbf{R}^{-1}\boldsymbol{j}\right\|^2 \\ &\leq \|\boldsymbol{j}\|^2 \end{aligned} \tag{14}$$

so that:

$$\frac{1}{T}\int_0^T f(t)^2\|\boldsymbol{\omega}(t)\|^2\, dt \leq J_\Sigma. \tag{15}$$

If multiple motion primitives exist that all achieve some high-level goal, this cost function may thus be used to rank the input aggressiveness of the primitives. The cost function is also computationally convenient, and closed form solutions for the optimal trajectories are given below.

C. Axes decoupling and trajectory generation

The nonlinear trajectory generation problem is simplified by decoupling the dynamics into three orthogonal inertial axes, and treating each axis as a triple integrator with jerk used as control input. The true control inputs f and $\boldsymbol{\omega}$ are then recovered from the jerk inputs using (9) and (12). The final state is determined from the goal end state components $\hat{\sigma}_i$ relevant to each axis, and the duration T is given.

The cost function of the three dimensional motion, J_Σ, is decoupled into a per-axis cost J_k by expanding the integrand in (13).

$$J_\Sigma = \sum_{k=1}^{3} J_k, \quad \text{where } J_k = \frac{1}{T}\int_0^T j_k(t)^2\,dt \tag{16}$$

For each axis k, the state $s_k = (p_k, v_k, a_k)$ is introduced, consisting of the scalars position, velocity, and acceleration. The jerk j_k is taken as input, such that

$$\dot{s}_k = f_s(s_k, j_k) = (v_k, a_k, j_k)\,. \tag{17}$$

Note again that the input constraints of Section II-A are not considered here, during the planning stage, but are deferred to Sections IV and VI.

For the sake of readability, the axis subscript k will be discarded for the remainder of this section where only a single axis is considered. The time argument t will similarly be neglected where it is considered unambiguous.

The optimal state trajectory can be solved straightforwardly with Pontryagin's minimum principle (see e.g. [25]) by introducing the costate $\lambda = (\lambda_1, \lambda_2, \lambda_3)$ and defining the Hamiltonian function $H(s, j, \lambda)$:

$$\begin{aligned} H(s,j,\lambda) &= \frac{1}{T}j^2 + \lambda^T f_s(s,j) \\ &= \frac{1}{T}j^2 + \lambda_1 v + \lambda_2 a + \lambda_3 j \qquad (18) \\ \dot{\lambda} &= -\nabla_s H(s^*, j^*, \lambda) = (0, -\lambda_1, -\lambda_2) \qquad (19) \end{aligned}$$

where j_k^* and s_k^* represent the optimal input and state trajectories, respectively.

The costate differential equation (19) is easily solved, and for later convenience the solution is written in the constants α, β and γ, such that

$$\lambda(t) = \frac{1}{T}\begin{bmatrix} -2\alpha \\ 2\alpha t + 2\beta \\ -\alpha t^2 - 2\beta t - 2\gamma \end{bmatrix}. \tag{20}$$

The optimal input trajectory is solved for as

$$\begin{aligned} j^*(t) &= \arg\min_{j(t)} H(s^*(t), j(t), \lambda(t)) \\ &= \frac{1}{2}\alpha t^2 + \beta t + \gamma \end{aligned} \tag{21}$$

from which the optimal state trajectory follows from integration of (17):

$$s^*(t) = \begin{bmatrix} \frac{\alpha}{120}t^5 + \frac{\beta}{24}t^4 + \frac{\gamma}{6}t^3 + \frac{a_0}{2}t^2 + v_0 t + p_0 \\ \frac{\alpha}{24}t^4 + \frac{\beta}{6}t^3 + \frac{\gamma}{2}t^2 + a_0 t + v_0 \\ \frac{\alpha}{6}t^3 + \frac{\beta}{2}t^2 + \gamma t + a_0 \end{bmatrix} \tag{22}$$

with the integration constants set to satisfy the initial condition $s(0) = (p_0, v_0, a_0)$.

The remaining unknowns α, β and γ are solved for as a function of the desired end translational variable components $\hat{\sigma}_i$ as given in (7).

1) Fully defined end translational state: Let the desired end position, velocity, and acceleration along this axis be $s(T) = (p_f, v_f, a_f)$, given by the components $\hat{\sigma}_i$. Then the unknowns α, β and γ are isolated by reordering (22):

$$\begin{bmatrix} \frac{1}{120}T^5 & \frac{1}{24}T^4 & \frac{1}{6}T^3 \\ \frac{1}{24}T^4 & \frac{1}{6}T^3 & \frac{1}{2}T^2 \\ \frac{1}{6}T^3 & \frac{1}{2}T^2 & T \end{bmatrix}\begin{bmatrix} \alpha \\ \beta \\ \gamma \end{bmatrix} = \begin{bmatrix} \Delta p \\ \Delta v \\ \Delta a \end{bmatrix} \tag{23}$$

where

$$\begin{bmatrix} \Delta p \\ \Delta v \\ \Delta a \end{bmatrix} = \begin{bmatrix} p_f - p_0 - v_0 T - \frac{1}{2}a_0 T^2 \\ v_f - v_0 - a_0 T \\ a_f - a_0 \end{bmatrix}. \tag{24}$$

Solving for the unknown coefficients yields

$$\begin{bmatrix} \alpha \\ \beta \\ \gamma \end{bmatrix} = \frac{1}{T^5}\begin{bmatrix} 720 & -360T & 60T^2 \\ -360T & 168T^2 & -24T^3 \\ 60T^2 & -24T^3 & 3T^4 \end{bmatrix}\begin{bmatrix} \Delta p \\ \Delta v \\ \Delta a \end{bmatrix}. \tag{25}$$

Thus, generating a motion primitive only requires evaluating the above matrix multiplication for each axis, after which the state along the primitive is found by evaluating (22).

2) Partially defined end translational state: Components of the final state may be left free by $\hat{\sigma}$. These states may correspondingly be specified as free when solving the optimal input trajectory, by noting that the corresponding costates must equal zero at the end time [25]. The closed form solutions to the six different combinations of partially defined end states are given in Appendix A – in each case solving the coefficients reduces to evaluating a matrix product.

3) Motion primitive cost: The per-axis cost value of (16) can be explicitly calculated as below. This is useful if multiple different candidate motion primitives are evaluated to achieve some higher-level goal. In this case, the primitive with the lowest cost can be taken as the 'least aggressive' in the sense of (14).

$$\begin{aligned} J =& \gamma^2 + \beta\gamma T + \frac{1}{3}\beta^2 T^2 + \frac{1}{3}\alpha\gamma T^2 \\ &+ \frac{1}{4}\alpha\beta T^3 + \frac{1}{20}\alpha^2 T^4 \end{aligned} \tag{26}$$

Note that this cost holds for all combinations of end translational variables.

IV. Determining feasibility

The motion primitives generated in the previous section did not take the input feasibility constraints of Section II-A1 into account – this section revisits these and provides computationally inexpensive tests for the feasibility/infeasibility of a given motion primitive with respect to the input constraints of (4) and (5). This section also provides a computationally inexpensive method to calculate the extrema of an affine combination of the translational variables along the primitive, allowing to test constraints of the form (8). In Section VI conditions are given under which feasible motion primitives are guaranteed to exist.

A. Input feasibility

Given some time interval $\mathcal{T} = [\tau_1, \tau_2] \subseteq [0, T]$ and three triple integrator trajectories of the form (22) with their corresponding jerk inputs $j_k(t)$, the goal is to determine whether corresponding inputs to the true system f and $\boldsymbol{\omega}$, as used in (1) and (2), satisfy feasibility requirements of Section II-A1. The choice of $\mathcal{T}$ is revisited when describing the recursive implementation, below. The tests are designed with a focus on computational speed, and provide sufficient, but not necessary, conditions for both feasibility and infeasibility – meaning that some motion primitives will be indeterminable with respect to these tests.

1) Thrust: The interval $\mathcal{T}$ is feasible with respect to the thrust limits (4) if and only if

$$\max_{t\in\mathcal{T}} f(t)^2 \leq f_{\max}^2 \text{ and} \tag{27}$$

$$\min_{t\in\mathcal{T}} f(t)^2 \geq f_{\min}^2. \tag{28}$$

Squaring (9) yields

$$f^2 = \|\ddot{\boldsymbol{x}} - \boldsymbol{g}\|^2 = \sum_{k=1}^{3} (\ddot{x}_k - g_k)^2 \tag{29}$$

where g_k is the component of gravity in axis k. Combining (27)-(29), the thrust constraints can be interpreted as spherical constraints on the acceleration.

By taking the per-axis extrema of (29) the below bounds follow.

$$\max_{t\in\mathcal{T}} (\ddot{x}_k(t) - g_k)^2 \leq \max_{t\in\mathcal{T}} f(t)^2 \text{ for } k \in \{1,2,3\} \tag{30}$$

$$\max_{t\in\mathcal{T}} f(t)^2 \leq \sum_{k=1}^{3} \max_{t\in\mathcal{T}} (\ddot{x}_k(t) - g_k)^2 \tag{31}$$

$$\min_{t\in\mathcal{T}} f(t)^2 \geq \sum_{k=1}^{3} \min_{t\in\mathcal{T}} (\ddot{x}_k(t) - g_k)^2 \tag{32}$$

These bounds will be used as follows: if the left hand side of (30) is greater than $f_{\max}$, the interval is definitely infeasible, while if both the right hand side of (31) is less than $f_{\max}$ and the right hand side of (32) is greater than $f_{\min}$, the interval is definitely feasible with respect to the thrust constraints.

Note that the value $\ddot{x}_k - g_k$ as given by (22) is a third order polynomial in time, meaning that its maximum and minimum (denoted $\bar{\ddot{x}}_k$ and $\underline{\ddot{x}}_k$, respectively) can be found in closed form by solving for the roots of a quadratic and evaluating $\ddot{x}_k - g_k$ at at most two points strictly inside $\mathcal{T}$, and at the boundaries of $\mathcal{T}$. The extrema of $(\ddot{x}_k - g_k)^2$ then follow as

$$\max_{t\in\mathcal{T}} (\ddot{x}_k(t) - g_k)^2 = \max\{\bar{\ddot{x}}_k^2, \underline{\ddot{x}}_k^2\} \tag{33}$$

$$\min_{t\in\mathcal{T}} (\ddot{x}_k(t) - g_k)^2 = \begin{cases} \min\{\bar{\ddot{x}}_k^2, \underline{\ddot{x}}_k^2\} & \text{if } \bar{\ddot{x}}_k \cdot \underline{\ddot{x}}_k \geq 0 \\ 0 & \text{otherwise,} \end{cases} \tag{34}$$

where $\bar{\ddot{x}}_k \cdot \underline{\ddot{x}}_k < 0$ implies a sign change (and thus a zero crossing) of $\ddot{x}_k(t) - g_k$ in $\mathcal{T}$.

Thus, from (30), a sufficient criterion for input infeasibility of the section is if

$$\max\{\bar{\ddot{x}}_k^2, \underline{\ddot{x}}_k^2\} > f_{\max}. \tag{35}$$

Similarly, from (31)-(32), a sufficient criterion for feasibility is if both

$$\sum_{k=1}^{3} \max\{\bar{\ddot{x}}_k^2, \underline{\ddot{x}}_k^2\} \leq f_{\max} \text{ and} \tag{36}$$

$$\sum_{k=1}^{3} \min\{\bar{\ddot{x}}_k^2, \underline{\ddot{x}}_k^2\} \geq f_{\min} \tag{37}$$

hold. If neither criterion (35) nor (36)-(37) applies, the section is marked as indeterminate with respect to thrust feasibility.

2) Body rates: The magnitude of the body rates can be bounded as a function of the jerk and thrust as below:

$$\omega_1^2 + \omega_2^2 \leq \frac{1}{f^2} \|j\|^2. \tag{38}$$

This follows from squaring (12), and using the following induced norm:

$$\left\| \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} \right\| \leq 1. \tag{39}$$

The right hand side of (38) can be bounded from above by $\bar{\omega}^2$, defined as below, which then also provides an upper bound for the sum $\omega_1^2 + \omega_2^2$. The terms in the denominator are evaluated as in (34).

$$\omega_1^2 + \omega_2^2 \leq \frac{1}{f^2} \|j\|^2 \leq \bar{\omega}^2 = \frac{\sum_{k=1}^{3} \max_{t\in\mathcal{T}} j_k(t)^2}{\sum_{k=1}^{3} \min_{t\in\mathcal{T}} (\ddot{x}_k(t) - g_k)^2} \tag{40}$$

Using the above equation, and assuming $\omega_3 = 0$, the time interval $\mathcal{T}$ can be marked as feasible w.r.t. the body rate input if $\bar{\omega}^2 \leq \omega_{\max}^2$, otherwise the section is marked as indeterminate.

3) Recursive implementation: The feasibility of a given time interval $\mathcal{T} \subseteq [0, T]$ is tested by applying the above two tests on $\mathcal{T}$. If both tests return feasible, $\mathcal{T}$ is input feasible and the algorithm terminates; if one of the tests returns infeasible, the algorithm terminates with the motion over $\mathcal{T}$ marked as input infeasible. Otherwise, the section is divided in half, such that

$$\tau_{\frac{1}{2}} = \frac{\tau_1 + \tau_2}{2} \tag{41}$$

$$\mathcal{T}_1 = [\tau_1, \tau_{\frac{1}{2}}], \quad \mathcal{T}_2 = [\tau_{\frac{1}{2}}, \tau_2]. \tag{42}$$

If the time interval $\tau_{\frac{1}{2}} - \tau_1$ is smaller than some user defined minimum $\underline{\Delta\tau}$, the algorithm terminates with the primitive marked indeterminable. Otherwise, the algorithm is recursively applied first to $\mathcal{T}_1$: if the result is feasible, the algorithm is recursively applied to $\mathcal{T}_2$. If $\mathcal{T}_2$ also terminates as a feasible section, the entire primitive can be marked as feasible, otherwise the primitive is rejected as infeasible/indeterminable. Thus, the test returns one of three outcomes:

- the inputs are provably feasible,
- the inputs are provably infeasible, or
- the tests were indeterminate, feasibility could not be established.

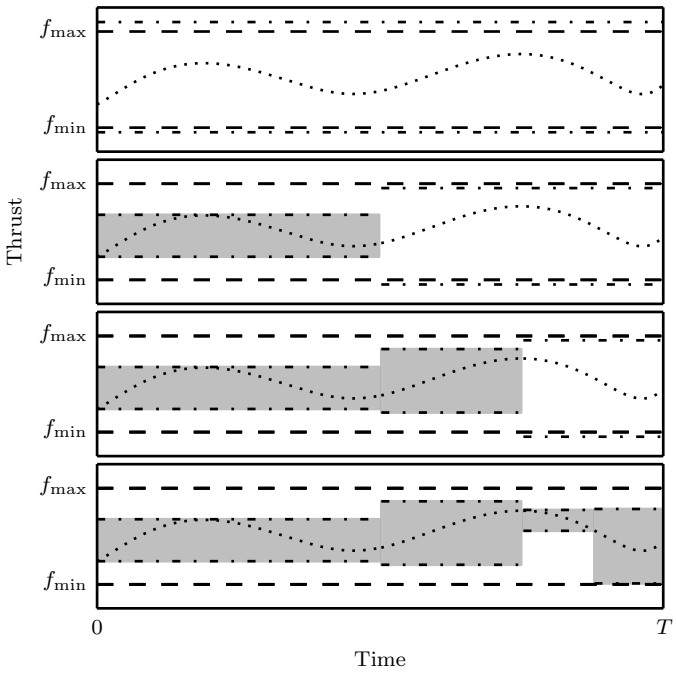


Fig. 3. Visualisation of the recursive implementation of the thrust feasibility test: The vehicle thrust limits $f_{\min}$ and $f_{\max}$ are shown as dashed lines, and the thrust trajectory is the dotted curve. First, the minimum and maximum bounds of (36) and (37) are calculated for the entire motion primitive, shown as the dash-dotted lines in the top plot. Because these bounds exceed the vehicle limits, the section is halved, and the tests are applied to each half (second plot). The left hand side section's bounds are now within the limits, so this section is marked feasible with respect to the thrust bounds (shown shaded in the plot), while the second half is again indeterminate. This is halved again, in the third plot, and a section is yet again halved in the fourth plot. Now, all sections are feasible with respect to the thrust limits, and the test terminates. Note that, in the implementation, the thrust infeasibility test of (35) and the body rate feasibility test (40) will be done simultaneously.

Note that the interval upper bound of (33) is monotonically nonincreasing with decreasing length of the interval $\mathcal{T}$, and similarly the lower bound of (34) is monotonically nondecreasing.

Furthermore, note that as $\tau_{\frac{1}{2}} - \tau_1$ tends to zero the right hand sides of (31)-(32) converge and the thrust feasibility test becomes exact. This does not, however, apply to the body rate feasibility test due to the induced norm in (39).

The recursive implementation of the sufficient thrust feasibility tests of (36)-(37) is visualised for an example motion primitive in Fig. 3.

4) Remark on convex approximations of thrust-feasible region: The feasible acceleration space of the vehicle follows from (27)-(29), and is non-convex due to the positive minimum thrust value. This is visualised in Fig. 4. Nonetheless, in the limit, the presented recursive thrust input test allows for testing feasibility over the entire thrust feasible space. This is an advantage when compared to approaches that require the construction of convex approximations of the feasible space (e.g. [20]). Consider, for example, a trajectory that begins with zero acceleration and ends with a final acceleration of $-2\boldsymbol{g}$ (i.e. the quadrocopter is upside down at the end of the manoeuvre) – such an example is shown in Fig. 4. The straight line connecting the initial and final accelerations crosses through the acceleration infeasible zone due to minimum thrust. Therefore no convex approximation can be constructed in the acceleration space in which to evaluate this trajectory.

B. Affine translational variable constraints

Referring to (22), it can be seen that calculating the range of some linear combination of the system's translational variables $\sigma(t)$ (of the form (8)) can be done by solving for the extrema of a polynomial of order at most five. This involves finding the roots of its derivative (a polynomial of order at most four) for which closed form solutions exist [26].

This is useful, for example, to verify that the quadrocopter does not fly into the ground, or that the position of the quadrocopter remains within some allowable flight space. Specifically, any planar position constraint can be specified by specifying that the inner product of the quadrocopter's position with the normal of the plane is greater than some constant value. It can also be used to calculate a bound on the vehicle's maximum velocity, which could be useful in some computer vision applications (e.g. [27]). Furthermore, an affine bound on the vehicle's acceleration can be interpreted as a bound on the tilt of the quadrocopter's $\boldsymbol{e}_3$ axis, through (1).

V. Choice of coordinate system

Because the Euclidean norm used in the cost (13) is invariant under rotation, the optimal primitive for some given problem will be the same when the problem is posed in any inertial frame, despite the axes being solved for independently of one another.

The Euclidean norm is also used in the feasibility tests of Section IV. In the limit, as the length of the time interval $\mathcal{T}$ tends to zero, both the thrust and body rates feasibility tests become invariant under rotation, and thus independent of the choice of coordinate system. The affine constraints of Section IV-B can be trivially transformed from one coordinate system to another, such that there exists no preferred coordinate system for a set of constraints. This allows the designer the freedom to pose the motion primitive generation problem in the most convenient inertial coordinate system.

VI. Guaranteeing feasibility

For some classes of trajectory, the existence of a feasible motion primitive can be guaranteed a priori, without running the tests described in the preceding section.

For the specific case of primitives starting and ending at rest (zero velocity, zero acceleration, and a given end point), a bound on the end time T will be explicitly calculated in dependence of the translation distance, such that any end time larger than this bound is guaranteed to be feasible with respect to the input constraints. Furthermore, the position trajectory for such rest-to-rest primitives is shown to remain on the straight line segment between the initial and final positions. Thus, given a convex allowable flight space, all primitives that start and end at rest and within the allowable flight space will remain within the allowable flight space for the entire trajectory.

The input feasibility of general motion primitives, with arbitrary initial and final accelerations and velocities, is also briefly

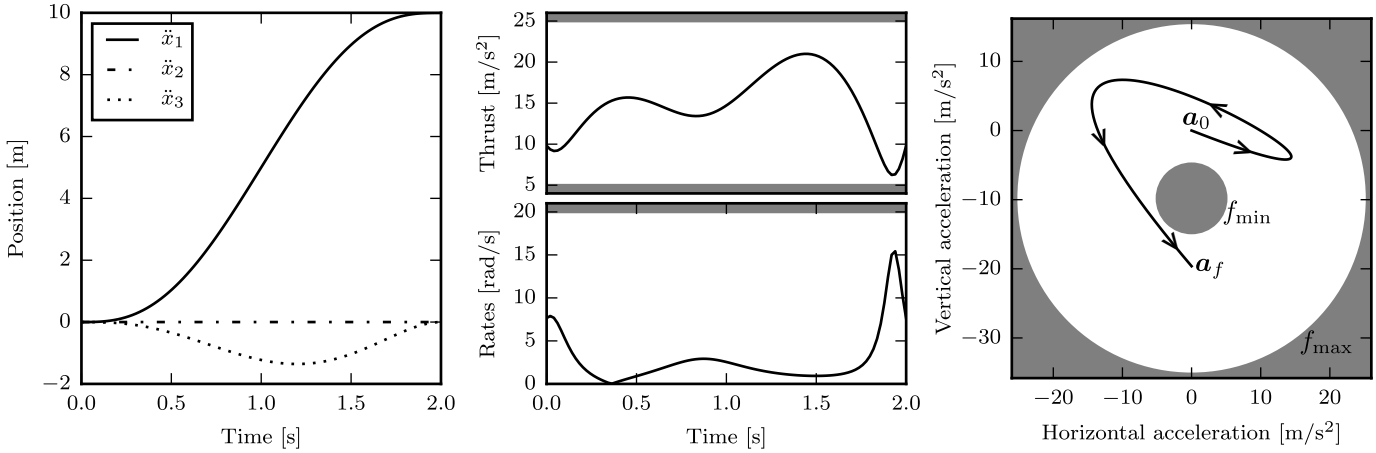


Fig. 4. Example trajectory demonstrating non-convexity of the feasible acceleration space. The trajectory moves an initially stationary quadrocopter 10 m horizontally, ending at zero velocity and with final acceleration $\boldsymbol{a}_f = -2\boldsymbol{g}$ in 2 s (i.e. the quadrocopter is upside down at the end). The left-most plot shows the position trajectory of the vehicle, the middle plots show the inputs during the trajectory (with the shaded regions being infeasible). The right-hand side plot shows the acceleration trajectory in the acceleration space, where the two concentric circles represent the minimum and maximum thrust limits (and are centred at $\boldsymbol{g}$) for $\ddot{x}_2 = 0$. The axes are chosen such that x_3 points opposite to gravity, and there is no motion along x_2. Note the non-convexity of the feasible acceleration space.

discussed. It should be noted that those motion primitives which can be guaranteed to be feasible a priori will be a conservative subset of the primitives which can be shown to be feasible a posteriori using the recursive tests described in Section IV. Further discussion is provided in Section VII.

A. Rest-to-rest primitives

First, an upper bound on the lowest end time T at which a rest-to-rest primitive becomes feasible with respect to the input constraints is calculated in dependence of the translation distance. Without loss of generality it is assumed that the quadrocopter's initial position coincides with the origin, and the problem is posed in a reference frame such that the final position is given as $(p_f, 0, 0)$, with p_f being the distance from the origin to the final location, such that $p_f \geq 0$.

From (25) it is clear that the optimal position trajectory along x_2 and x_3 has zero position for the duration of the primitive, and therefore also zero acceleration and velocity.

1) Input feasibility: The acceleration trajectory along axis 1 is calculated by solving the motion coefficients with (25), and then substituting into (22), such that

$$\ddot{x}_1(t) = 60\frac{t}{T^3}p_f - 180\frac{t^2}{T^4}p_f + 120\frac{t^3}{T^5}p_f. \tag{43}$$

Introducing the variable $\xi \in [0, 1]$ such that $t = \xi T$, and substituting for the above yields

$$\ddot{x}_1(\xi T) = \frac{60p_f}{T^2}\left(\xi - 3\xi^2 + 2\xi^3\right) \tag{44}$$

for which the extrema lie at

$$\xi^* = \frac{1}{2} \pm \frac{\sqrt{3}}{6} \tag{45}$$

$$\ddot{x}_1(\xi^* T) = \mp\frac{10\sqrt{3}p_f}{3T^2}. \tag{46}$$

Exploiting the observation that $|\ddot{x}(\xi^* T)|$ decreases monotonically with increasing T, an upper bound $T_{f_{\min}}$ for the end time at which such a primitive becomes feasible with respect to the minimum thrust constraint can be calculated. This is done by substituting (46) for the sufficient criterion of (32), under the worst case assumption that the motion is aligned with gravity, i.e. the acceleration in the directions perpendicular to gravity are zero. Then

$$T_{f_{\min}} = \sqrt{\frac{10p_f}{\sqrt{3}\left(\|\boldsymbol{g}\| - f_{\min}\right)}}. \tag{47}$$

Similarly, an upper bound $T_{f_{\max}}$ for the end time at which the primitive becomes feasible with respect to the maximum thrust constraint can be calculated by substituting the maximum acceleration bound of (46) into the sufficient criterion of (31). Again, the worst case occurs when the motion is aligned with gravity, and the final time can be calculated as

$$T_{f_{\max}} = \sqrt{\frac{10p_f}{\sqrt{3}\left(f_{\max} - \|\boldsymbol{g}\|\right)}}. \tag{48}$$

Finally, an upper bound $T_{\omega_{\max}}$ for the end time at which the primitive satisfies the body rates constraint is calculated as follows. First, the jerk along axis 1 is solved for with (21), and $t = \xi T$ is substituted as before, to give

$$j_1(\xi T) = \frac{60p_f}{T^3}\left(6\xi^2 - 6\xi + 1\right). \tag{49}$$

This has extrema at $\xi \in \{0, \frac{1}{2}, 1\}$, and the maximum of $|j_1(\xi T)|$ occurs at $\xi^* \in \{0, 1\}$, so that

$$\max_{t\in[0,T]} |j_i(t)| = |j_i(\xi^* T)| = \frac{60p_f}{T^3}. \tag{50}$$

Again, this maximum is monotonically decreasing for increasing end time T.

This value is then substituted for the numerator of (40), and it is assumed that the primitive satisfies the minimum thrust constraint, so that $f_{\min}$ can be substituted for the denominator. This makes the conservative assumption that the maximum jerk value is achieved at the same time as the minimum thrust

value. Equating the result to $\omega_{\max}$, and solving for $T_{\omega_{\max}}$ yields

$$T_{\omega_{\max}} = \sqrt[3]{\frac{60 p_f}{\omega_{\max} f_{\min}}}. \tag{51}$$

Note that this requires $f_{\min}$ to be strictly positive (instead of non-negative as specified in (4)).

Combining (47), (48), and (51), any rest-to-rest primitive within a ball of radius p_f is guaranteed to be feasible with respect to the input constraints of Section II-A1 if the end time T is chosen to satisfy the below bound.

$$T \geq \max\{T_{f_{\min}}, T_{f_{\max}}, T_{\omega_{\max}}\}. \tag{52}$$

The conservatism of this bound is investigated in Section VII.

2) Position feasibility: It will be shown that the position along a rest-to-rest primitive remains on the straight line segment between the initial and final positions, independent of the end time T. Solving the full position trajectory as given in Section III-C1 and substituting $p_0 = 0$, $v_0 = v_f = 0$ and $a_0 = a_f = 0$, the position trajectory in axis 1 is given by

$$x_1(t) = p_f \left(6\frac{t^5}{T^5} - 15\frac{t^4}{T^4} + 10\frac{t^3}{T^3} \right) \tag{53}$$

with $p_f \geq 0$ the desired displacement along axis 1. Substituting again for $\xi = t/T$ such that $\xi \in [0, 1]$, the position trajectory can be rewritten as

$$x_1(\xi T) = p_f \left(6\xi^5 - 15\xi^4 + 10\xi^3 \right) \tag{54}$$

It is now straight forward to show that the extrema of $x_1(\xi T)$ are at $\xi^* \in \{0, 1\}$, and specifically that

$$x_1(t) \in [0, p_f]. \tag{55}$$

Axes 2 and 3 will remain at zero, so that the rest-to-rest primitive will travel along the straight line segment connecting the initial and final positions in three dimensional space. Therefore, given a convex allowable flight space, if the initial and final position are in the allowable flight space, all rest-to-rest primitives will remain within the allowable flight space.

3) Maximum velocity: In some applications it is desirable to limit the maximum velocity of the vehicle along the motion, most notably where the quadrocopter's pose is estimated with onboard vision [27]. For rest-to-rest primitives of duration T, the maximum speed occurs at $t = T/2$, and equals

$$\max_{t\in[0,T]} \|\dot{\boldsymbol{x}}(t)\| = \max_{t\in[0,T]} |\dot{x}_1(t)| = \frac{15}{8}\frac{p_f}{T}. \tag{56}$$

Thus, given some maximum allowable velocity magnitude and a distance to translate, the primitive end time T at which this maximum velocity will not be exceeded can be readily calculated from the above.

B. Guarantees for general primitives

For general primitives, with nonzero initial and/or final conditions, and with possibly unconstrained end states, it is harder to provide conditions under which feasible primitives can be guaranteed. Indeed, cases can be constructed which are input feasible for some specific end times, but become infeasible if the time is extended. It can however be stated for a motion primitive along an axis k (with arbitrary initial and final conditions and with any combination of final translational variable constraints in the goal state) that as the end time T tends to infinity:

- the magnitude of the jerk trajectory tends to zero, and
- the magnitude of the acceleration trajectory becomes independent of both initial and final position and velocity constraints, and can be bounded as

$$\lim_{T\to\infty} \max_{t\in[0,T]} \ddot{x}_k(t) \leq \max\{|a_{k0}|, |a_{kf}|\} \tag{57}$$

$$\lim_{T\to\infty} \min_{t\in[0,T]} \ddot{x}_k(t) \geq -\max\{|a_{k0}|, |a_{kf}|\}. \tag{58}$$

If the final acceleration is not specified, the acceleration bounds are

$$\lim_{T\to\infty} \max_{t\in[0,T]} \ddot{x}_k(t) \leq |a_{k0}| \tag{59}$$

$$\lim_{T\to\infty} \min_{t\in[0,T]} \ddot{x}_k(t) \geq -|a_{k0}|. \tag{60}$$

The calculations to show this can be found in Appendix B.

This knowledge can then be combined with the input feasibility constraints (similar to the rest-to-rest primitives, above), to guarantee the existence of an input feasible motion primitive based on the values of $|a_0|$ and $|a_f|$, at least as the end time is extended to infinity. Furthermore, by expanding the acceleration trajectory from (22) and applying the triangle inequality, the magnitude of the acceleration for finite end times can also be bounded. This bound can then be used to calculate an upper bound on the end time T at which the primitive will be feasible with respect to the inputs, however this bound will typically be very conservative.

ALGORITHM OVERVIEW

The focus in the preceding sections is on the generation and feasibility verification for a quadrocopter motion primitive, with an arbitrary initial state, to a set of goal end translational variables $\hat{\sigma}_i$ in a given time T. The resulting acceleration trajectory, along any axis, is a cubic polynomial in time. These trajectories minimize an upper bound representative of a product of the inputs, and computationally convenient methods are presented to test whether these trajectories are feasible with respect to input constraints, and with respect to bounds on linear combinations of the translational variables. Guarantees on feasible trajectories are given, with a specific focus on rest-to-rest trajectories.

In the following section the set of end times and goal end translational variables for which feasible trajectories can be found with the presented approach is compared to the total set of feasible trajectories, for the class of rest-to-rest trajectories. The computational cost of the presented approach is investigated in Section VIII. Section IX describes an experimental demonstration, where the presented primitives are incorporated into a larger trajectory generator.

VII. Conservatism

If the motion primitive computed with the presented approach is not feasible, it does not imply that no feasible trajectory is possible. There are two reasons for this:

- the trajectories generated in Section III are restricted to have accelerations described by cubic polynomials in time, and
- the feasibility verification of Section IV is sufficient, but not necessary.

This section attempts to give an indication of the space of end times T and end translational variables $\hat{\sigma}_i$ for which a quadrocopter could fly a trajectory, but the presented method is unable to find a feasible motion primitive. This section will specifically consider rest-to-rest motions.

In [28] an algorithm is given to compute minimum time trajectories which satisfy Pontryagin's minimum principle, for the quadrocopter system dynamics and input constraints of Section II-A. These trajectories represent the surface of the feasible region for quadrocopter rest-to-rest trajectories: given a desired final translation, a feasible trajectory exists with any end time larger than the time optimal end time (e.g. by executing the time optimal trajectory, and then simply waiting). By definition, no feasible trajectory exists for shorter end times.

Fig. 5 compares this feasible region to those trajectories that can be found with the methods of Section III and Section IV. The system limits were set as in [28], with $f_{\min} = 1\,\text{m/s}^2$, $f_{\max} = 20\,\text{m/s}^2$, $\omega_{\max} = 10\,\text{rad/s}$. For a given translation distance d, the desired end translational variables are defined such that all components are zero, except the position in the direction of motion which is set to d.

For each distance d, the space of feasible end times was determined as follows. A set of end times was generated, starting at zero and with increments of 1 ms. For each end time, a motion primitive was generated, and the set of end distances and end times (T, d) for which an input feasible trajectory was found is shown in Fig. 5. For the sake of comparison, the guaranteed feasible end time given in Section VI-A is also plotted.

The fastest feasible manoeuvre found with the presented algorithm requires on the order of 50% longer than the time optimal trajectory of [28] when translating vertically, and on the order of 20% longer when translating horizontally. From the figure it can be seen that the guaranteed feasible end time of Section VI is quite conservative, requiring the order of three times longer for the manoeuvre than the time optimal trajectory. However, these trajectories can be computed at very low cost, specifically requiring no iterations to determine feasibility.

VIII. Computation times

This section presents statistics for the computational cost of the presented algorithm when implemented on a standard laptop computer, and on a standard microcontroller. The algorithm was implemented in C++. Except for setting the compiler optimisation to maximum, no systematic attempt was made to optimise the code for speed. To evaluate the time

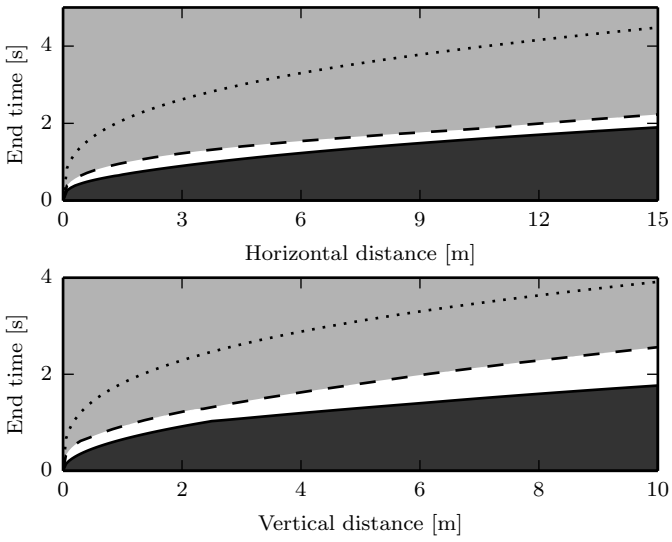


Fig. 5. The set of horizontal/vertical final displacements for which the presented algorithm is able to find input feasible trajectories (the lightly shaded area above the dashed line), and the set of final displacements and end times for which no trajectories are possible (the darkly shaded area, with the boundary as given by the time optimal trajectories of [28]). The dotted line is the conservative end time guarantee as calculated in Section VI-A. The white area represents displacements that could be reached by the vehicle, but where the presented method cannot find a feasible motion primitive.

required to compute the motion primitives described in this paper, primitives were generated for a quadrocopter starting at rest, at the origin, and translating to a final position chosen uniformly at random in a box with side length 4 m, centred at the origin. The target end velocity and acceleration were also chosen uniformly between $-2\,\text{m/s}$ and $2\,\text{m/s}$, and $-2\,\text{m/s}^2$ and $2\,\text{m/s}^2$, respectively. The end time was chosen uniformly at random between $0.2\,\text{s}$ and $10\,\text{s}$. The algorithm was implemented on a laptop computer with an Intel Core i7-3720QM CPU running at 2.6GHz, with 8GB of RAM, with the solver compiled with Microsoft Visual C++ 11.0. The solver was run as a single thread.

For one hundred million such motion primitives, the average computation time per primitive was measured to be $1.05\,\mu\text{s}$, or approximately 950'000 primitives per second. This includes

- generating the primitive,
- verifying that the inputs along the trajectory are feasible (with the recursive test of Section IV-A3, for $f_{min} = 5\,\text{m/s}^2$, $f_{max} = 25\,\text{m/s}^2$, and $\omega_{max} = 20\,\text{rad/s}$, and
- verifying that the position of the quadrocopter stays within a $4 \times 4 \times 4$m box centred at the origin (that is, six affine constraints of the form (8) as described in Section IV-B).

Of the primitives, 91.6% tested feasible with respect to the inputs, 6.4% infeasible, and the remaining 2.0% were indeterminate.

If the position feasibility test is not performed, the average computation time drops to $0.74\,\mu\text{s}$, again averaged over the generation of one hundred million random primitives, or approximately 1.3 million per second.

The algorithm was also implemented on the STM32-F4 microcontroller, which costs approximately €10 and is, for example, used on the open-source PX4 autopilot system [29].

One million random primitives were generated, with the same parameters as above. The average execution time was $149\,\mu s$ per primitive (or approximately 6'700 primitives per second), including trajectory generation, input feasibility, and position feasibility tests. If the position feasibility test is not performed, the average computation time drops to $95\,\mu s$ per primitive, or approximately 10'500 per second.

IX. Example application and experimental results

This section presents an example of a high level trajectory generator that uses the motion primitives to achieve a high level goal (as illustrated in Fig. 1). The high-level goal is for a quadrocopter, with an attached net, to catch a ball thrown by a person. This task was chosen due to its highly dynamic three-dimensional nature, the requirement for real-time trajectory generation, and the existence of a variety of trajectories which would achieve the goal of catching the ball. All the presented data is from actual flight experiments.

The algorithm is applied in a naïve brute force approach, to illustrate the ease with which a complex, highly dynamic, quadrocopter task may be encoded and solved. The multimedia attachment contains a video showing the quadrocopter catching the ball.

To catch the ball, the motion primitive generator is used in two different ways:

- to generate trajectories to a catching point, starting from the quadrocopter's current state and ending at a state and time at which the ball enters the attached net, and
- to generate stopping trajectories, which are used to bring the quadrocopter to rest after the ball enters the net.

The experiments were conducted in the Flying Machine Arena, a space equipped with an overhead motion capture system which tracks the pose of the quadrocopters and the position of the balls. A PC processes the motion capture data, and generates commands that are transmitted wirelessly to the vehicles at $50\,Hz$. More information on the system can be found in [30]. The quadrocopter used is a modified Ascending Technologies "Hummingbird" [31], with a net attached approximately $18\,cm$ above the vehicle's centre of mass (see Fig. 6).

Fig. 6. A quadrocopter with attached catching net, as used to demonstrate the algorithm. The centre of the net is mounted above the vehicle's centre of mass in the quadrocopter's $\boldsymbol{e}_3$ direction.

1) Catching trajectories: The computational speed of the presented approach is exploited to evaluate many different ways of catching the ball. This is done with a naïve brute force approach, where thousands of different primitives are generated at every controller step. Each primitive encodes a different strategy to catch the ball, of which infeasible primitives are rejected and the 'best' is kept from the remainder. This is done in real-time, and used as an implicit feedback law, with one controller cycle typically involving the evaluation of thousands of candidate motion primitives. This task is related to that of hitting a ball with a racket attached to a quadrocopter, as was previously investigated in [32] and [21].

The catching task is encoded in the format of Section II-B by stipulating that the centre of the net must be at the same position as the ball, and the velocity of the quadrocopter perpendicular to the ball's flight must be zero. The requirement on the velocity reduces the impact of timing errors on the system. Because the centre of the net is not at the centre of the quadrocopter, the goal end state must also include the quadrocopter's normal direction $\boldsymbol{e}_3$ – given some ball position, there exists a family of quadrocopter positions and normal directions which will place the centre of the net at the ball's position. The desired end translational variables $\hat{\sigma}$ thus contains the quadrocopter's position, its normal direction (thus acceleration), and its velocity components perpendicular to the ball's velocity (thus specifying eight of the nine possible variables). The strategy for specifying these eight variables is described below.

The ball is modelled as a point mass with quadratic drag, and at every controller update step its trajectory is predicted until it hits the floor. This is discretized to generate a set of possible catching times $T^{(k)}$, with the discretization time step size chosen such that at most 20 end times are generated, and that the time step size is no smaller than the controller update period.

For each of these possible catching times $T^{(k)}$, a set of candidate desired end translational variables $\hat{\sigma}^{(j,k)}$ is generated as follows. The quadrocopter's goal normal direction is generated by generating 49 candidate end normals, centred around the direction opposing the ball's velocity at time $T^{(k)}$. To convert these candidate end normals to goal accelerations it is necessary to further specify an end thrust value for each. The goal acceleration can then be calculated with (1). For each of the orientations generated, ten candidate final thrust values are used, spaced uniformly between $f_{\min}$ and $f_{\max}$.

Given an end normal direction, the required quadrocopter position at the catching instant can be calculated as that position placing the centre of the net at the ball (for the 49 different normal direction candidates, the end location of the vehicle centre of mass will be located on a sphere centred at the ball's position).

The quadrocopter's velocity is required to be zero in the directions perpendicular to the ball's velocity at the catching time, while the quadrocopter velocity component parallel to the ball's velocity is left free. Thus eight components of the end translational variables in (7) are specified.

For each of the candidate catching instants, there are 490 candidate end states to be evaluated. Because the number of

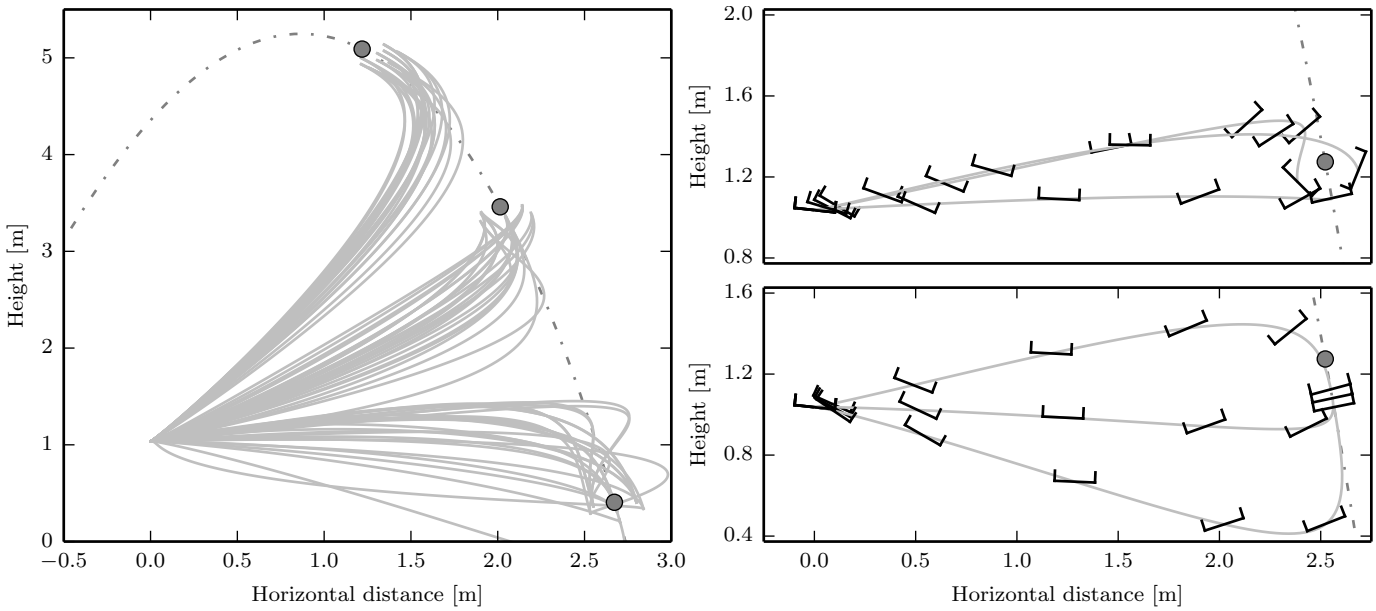


Fig. 7. Sampled motion primitives at one time step of a catching manoeuvre: On the left is shown the quadrocopter's centre of mass for 81 candidate primitives to catch a thrown ball. The primitives are shown as solid lines, starting at the position $(0, 1)$m, and the ball's predicted trajectory is shown as a dash-dotted line, moving from left to right in the figure. The ball's position at each of three candidate catching instants is shown as a solid circle. Each candidate primitive places the centre of the net at the ball, with the quadrocopter's velocity at that instant parallel to the ball's velocity. Note that the final orientation is a parameter that is searched over, as is the thrust value at the catching instant. A total of 2'812 candidates were evaluated at this time instant, which required 3.05 ms of computation. The primitives shown passing through the ground (at 0 m height) are eliminated when the position boundaries are evaluated. The right most two plots show detail of some of these primitives, showing the quadrocopter's orientation along the primitives. At the top-right plot, three candidate primitives are shown with different end orientations (and only showing orientations lying in the plane of the plot). The lower right plot shows primitives to the same orientation, but with varying end thrusts.

end times is limited to 20, this means that there are at most 9'800 catching primitives of the form $\left(T^{(k)}, \hat{\sigma}^{(j,k)}\right)$ calculated at any controller update step.

Next the candidate primitive is tested for feasibility with respect to the inputs as described in Section IV-A, with the input limits set to $f_{min} = 5\,\text{m/s}^2$, $f_{max} = 25\,\text{m/s}^2$ and $\omega_{max} = 20\,\text{rad/s}$. Then the candidate is tested for position feasibility, where the position trajectory is verified to remain inside a safe box as described in Section IV-B – this is to remove trajectories that would either collide with the floor or the walls. If the primitive fails either of these tests, it is rejected.

Some such candidate motion primitives are visualised in Fig. 7.

For each candidate catching primitive remaining, a stopping trajectory will be searched for (described in more detail below). If no stopping trajectory for a candidate catching primitive is found that satisfies both the input feasibility constraints and position constraints, that catching candidate is removed from the set.

Now, each remaining candidate is feasible with respect to input and position constraints, both for catching the ball, and the stopping manoeuvre after the ball is caught. From this set, that candidate with the lowest cost J_Σ (as defined in Section III-C3) is selected as the best.

2) Stopping trajectories: At the catching instant, a catching candidate trajectory will generally have a nonzero velocity and acceleration, making it necessary to generate trajectories from this state to rest. For these stopping trajectories, the goal end state translational variables specify that the velocity and acceleration must be zero, but leave the position unspecified. The primitive duration is sampled from a set of 6 possibilities, ranging from 2 down to 0.25 seconds. The search is terminated after the first stopping primitive is found that is feasible with respect to the inputs and the position constraints. Two such stopping primitives are shown in Fig. 8.

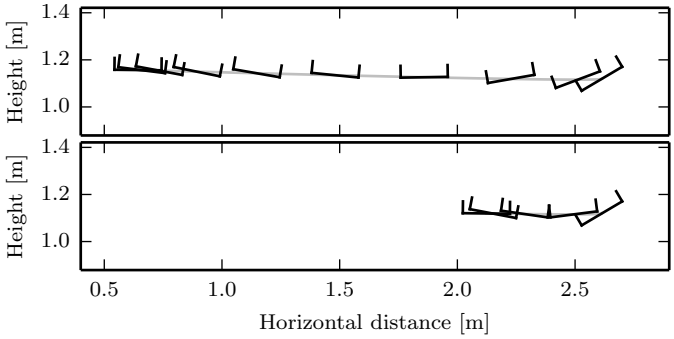


Fig. 8. Sampled stopping motion primitives: Two candidate stopping primitives bringing the quadrocopter to rest, starting at the catching instant of the optimal catching primitive from Fig. 7. The quadrocopter moves from right to left in both figures. The upper stopping candidate brings the quadrocopter to rest in 2 s, the lower in 1 s.

3) Closed loop control: Each remaining candidate catching primitive is feasible with respect to the input constraints, the position box constraints, and has a feasible stopping primitive. From these, the catching candidate with the lowest cost value is then selected as the best. This algorithm is then applied as an implicit feedback law, as in model predictive control [33], such that the entire process is repeated at the controller frequency of 50 Hz – thus the high level trajectory generator must run in

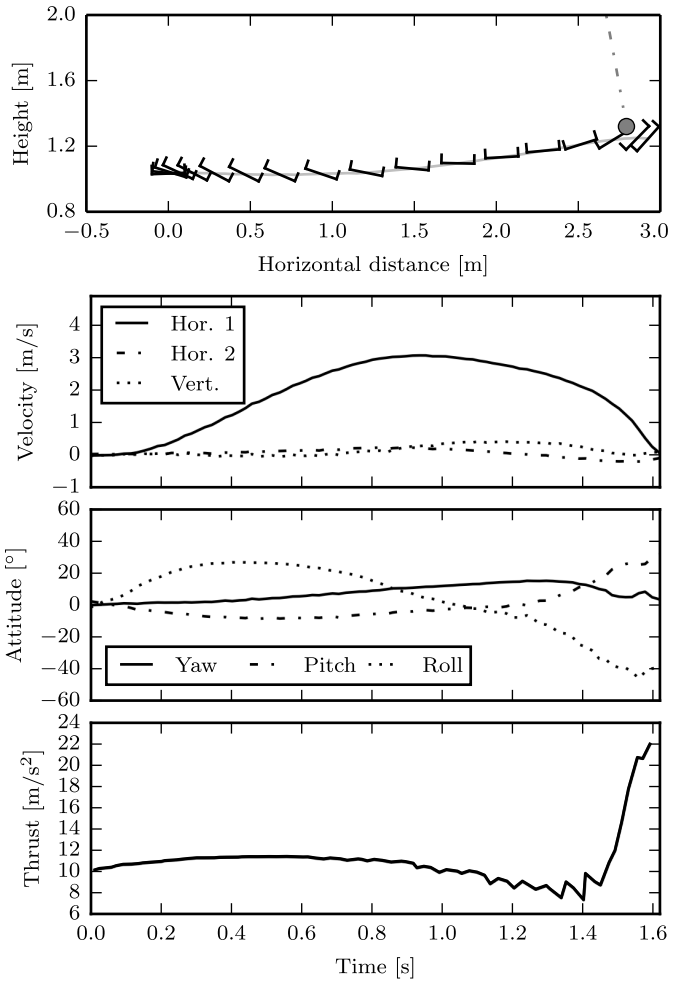


Fig. 9. The actual trajectory flown to catch the ball shown in Fig. 7. The top-most plot shows the ball's flight shown as a dash-dotted line, and the ball's position at the catching instant shown as a solid circle. Note that the offset between the vehicle and the ball at the catching instant is due to the net's offset from the quadrocopter's centre of mass. The three lower plots show the manoeuvre history, until the catching instant, with (from top to bottom) the quadrocopter's velocity, attitude, and thrust commands. The attitude of the vehicle is shown using the conventional 3-2-1 Euler yaw-pitch-roll sequence [24]. The bottom plot shows the thrust command, which can be seen to be within the limits of 5 to 25 $\mathrm{m\,s^{-2}}$. It should be noted that the motion primitives are applied as an implicit feedback law, and thus the flown trajectory does not correspond to any single planned motion primitive.

under 20 ms. This allows the system to implicitly update the trajectories as the prediction of the ball's flight becomes more precise, as well as compensate for disturbances acting on the quadrocopter.

If no candidate catching primitive remains, the last feasible trajectory is used as reference trajectory, tracked under feedback control. This typically happens at the end of the motion, as the end time goes to zero. After the ball is caught, the stopping primitive is executed. The stopping primitive is used as a reference trajectory tracked by the controller described in [30].

The completed catching trajectory corresponding to the candidates of Fig. 7 is shown in Fig. 9. The catching manoeuvre lasted 1.63 s, during which a total of 375'985 motion primitives were evaluated (including both catching and stopping manoeuvres). To catch the ball, the quadrocopter translated a distance of 2.93 m, having started at rest.

The attached video shows that the quadrocopter manages to catch thrown balls, validating the brute-force approach used to encode this problem. The video also shows the acrobatic nature of the resulting manoeuvres.

X. Conclusion

This paper presents a motion primitive that is computationally efficient both to generate, and to test for feasibility. The motion primitive starts at an arbitrary quadrocopter state, and generates a thrice differentiable position trajectory guiding the quadrocopter to a set of desired end translational variables (consisting of any combination of components of the vehicle's position, velocity, and acceleration). The acceleration allows to encode two components of the vehicle's attitude. The time to calculate a motion primitive and apply input and translational feasibility tests is shown to be on the order of a microsecond on a standard laptop computer.

The algorithm is experimentally demonstrated by catching a thrown ball with a quadrocopter, where it is used as part of an implicit feedback controller. In the application, thousands of candidate primitives are calculated and evaluated per controller update step, allowing the search over a large space of possible catching manoeuvres.

The algorithm appears well-suited to problems requiring to search over a large trajectory space, such as probabilistic planning algorithms [34], or the problem of planning for dynamic tasks with multiple vehicles in real time, similar to [35]. The algorithm may be especially useful if the high-level goal is described by non-convex constraints.

The presented motion primitive is independent of the quadrocopter's rotation about it's thrust axis (the Euler yaw angle). This is reflected in that the resulting commands do not specify a yaw rate, ω_3. A useful extension would be to compute an input trajectory for the quadrocopter yaw rate to achieve a desired final yaw angle.

It is shown that constraints on affine combinations of the quadrocopter's position, velocity, and acceleration may be tested efficiently. An interesting extension would be to investigate efficient tests for alternative constraint sets, for example more general convex sets.

Implementations of the algorithm in both Python and C++ can be found at [19].

Appendix A
Solutions for different end states constraints

Here the solutions for each combination of fixed end state is given in closed form for one axis. The states can be recovered by evaluating (22), and the trajectory cost from (26). The values Δp, Δv and Δa are calculated by (24).

Fixed position, velocity, and acceleration

$$\begin{bmatrix}\alpha\\ \beta\\ \gamma\end{bmatrix} = \frac{1}{T^5}\begin{bmatrix}720 & -360T & 60T^2\\ -360T & 168T^2 & -24T^3\\ 60T^2 & -24T^3 & 3T^4\end{bmatrix}\begin{bmatrix}\Delta p\\ \Delta v\\ \Delta a\end{bmatrix} \tag{61}$$

Fixed position and velocity

$$\begin{bmatrix}\alpha\\ \beta\\ \gamma\end{bmatrix} = \frac{1}{T^5}\begin{bmatrix}320 & -120T\\ -200T & 72T^2\\ 40T^2 & -12T^3\end{bmatrix}\begin{bmatrix}\Delta p\\ \Delta v\end{bmatrix} \tag{62}$$

Fixed position and acceleration

$$\begin{bmatrix}\alpha\\ \beta\\ \gamma\end{bmatrix} = \frac{1}{2T^5}\begin{bmatrix}90 & -15T^2\\ -90T & 15T^3\\ 30T^2 & -3T^4\end{bmatrix}\begin{bmatrix}\Delta p\\ \Delta a\end{bmatrix} \tag{63}$$

Fixed velocity and acceleration

$$\begin{bmatrix}\alpha\\ \beta\\ \gamma\end{bmatrix} = \frac{1}{T^3}\begin{bmatrix}0 & 0\\ -12 & 6T\\ 6T & -2T^2\end{bmatrix}\begin{bmatrix}\Delta v\\ \Delta a\end{bmatrix} \tag{64}$$

Fixed position

$$\begin{bmatrix}\alpha\\ \beta\\ \gamma\end{bmatrix} = \frac{1}{T^5}\begin{bmatrix}20\\ -20T\\ 10T^2\end{bmatrix}\Delta p \tag{65}$$

Fixed velocity

$$\begin{bmatrix}\alpha\\ \beta\\ \gamma\end{bmatrix} = \frac{1}{T^3}\begin{bmatrix}0\\ -3\\ 3T\end{bmatrix}\Delta v \tag{66}$$

Fixed acceleration

$$\begin{bmatrix}\alpha\\ \beta\\ \gamma\end{bmatrix} = \frac{1}{T}\begin{bmatrix}0\\ 0\\ 1\end{bmatrix}\Delta a \tag{67}$$

APPENDIX B
DERIVATION OF ACCELERATION BOUNDS

In Section VI-B the claim is made that the acceleration can be bounded as the end time T tends to infinity. This is shown here for each of the different combination of end translational variable constraints. The constraints will be divided into those that constrain the final acceleration, and those that do not.

A. Constrained final acceleration

If the final acceleration is specified, it will be shown that the acceleration can be bounded as in (57)-(58).

1) Fixed position, velocity, and acceleration: Substituting for the parameters α, β, and γ from (61) into (22), the acceleration trajectory for a fully specified set of end translational variables is

$$\begin{aligned}\ddot{x}(t) =& a_0 - \frac{9a_0}{T}t + \frac{3a_f}{T}t + \frac{18a_0}{T^2}t^2 - \frac{12a_f}{T^2}t^2\\ &- \frac{36t}{T^2}v_0 - \frac{24t}{T^2}v_f - \frac{10a_0}{T^3}t^3 + \frac{10a_f}{T^3}t^3\\ &- \frac{60p_0}{T^3}t + \frac{60p_f}{T^3}t + \frac{96v_0}{T^3}t^2 + \frac{84v_f}{T^3}t^2\\ &+ \frac{180p_0}{T^4}t^2 - \frac{180p_f}{T^4}t^2 - \frac{60v_0}{T^4}t^3 - \frac{60v_f}{T^4}t^3\\ &- \frac{120p_0}{T^5}t^3 + \frac{120p_f}{T^5}t^3.\end{aligned} \tag{68}$$

Introducing the variable $\xi := t/T \in [0,1]$ and letting $T \to \infty$, yields

$$\begin{aligned}\lim_{T\to\infty}\ddot{x}(\xi T) =& -10a_0\xi^3 + 18a_0\xi^2 - 9a_0\xi\\ &+ a_0 + 10a_f\xi^3 - 12a_f\xi^2 + 3a_f\xi.\end{aligned} \tag{69}$$

The above does not contain the initial and final position or velocity, as was claimed in Section VI-B. This means that in the limit as the duration T tends to infinity, the acceleration trajectory becomes independent of the position and velocity, for a fully defined end state. Next, it will be shown that (57) and (58) hold.

To do this, three cases will be examined independently:

- Case 1: $a_0 = a_f = 0$,
- Case 2: $|a_f| \leq |a_0|$, and
- Case 3: $|a_f| > |a_0|$.

For Case I, trivially, the right hand side of (69) goes to zero, such that

$$\lim_{T\to\infty}\ddot{x}(\xi T) = 0. \tag{70}$$

For Case II, we define the ratio $\rho_2 = a_f/a_0 \in [-1,1]$. It will be shown that the ratio between the maximum acceleration along the trajectory and the initial acceleration a_0 is in the range $[-1,1]$. Substituting into (69), yields the acceleration ratio as a function $\phi_2(\xi,\rho_2)$ of two variables:

$$\begin{aligned}\phi_2(\xi,\rho_2) :=& \lim_{T\to\infty}\frac{\ddot{x}(\xi T)|_{a_f=\rho_2 a_0}}{a_0}\\ =& \xi^3(10\rho_1 - 10) + \xi^2(-12\rho_1 + 18)\\ &+ 3\xi(\rho_1 - 3) + 1\end{aligned} \tag{71}$$

The extrema of ϕ_2 over $\xi \in [0,1]$ and $\rho_2 \in [-1,1]$ can be calculated straight-forwardly, and the minimum and maximum value of ϕ_2 calculated. From this can be calculated

$$\phi_2(\xi,\rho_2) \in [-1,1] \tag{72}$$

from which then follows that $\forall\xi \in [0,1], |a_f| \leq |a_0|$:

$$\lim_{T\to\infty}|\ddot{x}(\xi T)| \leq \max\{|a_0|, |a_f|\}. \tag{73}$$

For Case III, we define the ratio $\rho_3 = a_0/a_f \in (-1,1)$. It will be shown that the ratio between the acceleration extrema along the trajectory and the acceleration a_f is in the range $[-1,1]$.

Substituting into (69), again yields the acceleration ratio as a function of two variables $\phi_3(\xi,\rho_3)$:

$$\begin{aligned}\phi_3(\xi,\rho_3) :=& \lim_{T\to\infty}\frac{\ddot{x}(\xi T)|_{a_0=\rho_3 a_f}}{a_f}\\ =& \rho_2 + \xi^3(-10\rho_2 + 10) + \xi^2(18\rho_2 - 12)\\ &- 3\xi(3\rho_2 - 1)\end{aligned} \tag{74}$$

Similar to case 2 before, the extrema of ϕ_3 over $\xi \in [0,1]$ and $\rho_3 \in [-1,1]$ can be calculated. Note that the closed interval is used for ρ_3, for convenience. Then

$$\phi_3(\xi,\rho_3) \in [-1,1] \tag{75}$$

from which follows that $\forall\xi \in [0,1], |a_f| > |a_0|$:

$$\lim_{T\to\infty}|\ddot{x}(\xi T)| \leq \max\{|a_0|, |a_f|\}. \tag{76}$$

2) Fixed velocity and acceleration: If only the velocity and acceleration are fixed, the same procedure can be used as above, specifically evaluating the same three cases. Analogously to (68) the acceleration trajectory is

$$\begin{aligned}\ddot{x}(t) =& a_0 - \frac{4a_0}{T}t - \frac{2a_f}{T}t + \frac{3a_0}{T^2}t^2 + \frac{3a_f}{T^2}t^2 \\ &- \frac{6t}{T^2}v_0 + \frac{6t}{T^2}v_f + \frac{6v_0}{T^3}t^2 - \frac{6v_f}{T^3}t^2.\end{aligned} \tag{77}$$

Analogously to (69) the limit can be taken, and the variable ξ introduced:

$$\lim_{T\to\infty} \ddot{x}(\xi T) = 3a_0\xi^2 - 4a_0\xi + a_0 + 3a_f\xi^2 - 2a_f\xi \tag{78}$$

Applying the same three cases as before, (57)-(58) follow.

3) Fixed acceleration: If only the end acceleration is specified, the acceleration trajectory is

$$\ddot{x}(t) = a_0 - \frac{a_0 t}{T} + \frac{a_f t}{T}. \tag{79}$$

From this, and noting that $t/T \in [0,1]$, (57)-(58) follow trivially.

B. Unconstrained final acceleration

If the final acceleration is not fixed, the procedure of Section B-A1 must be modified slightly.

1) Fixed position and velocity: Analogously to (69) the acceleration in this case is

$$\begin{aligned}\ddot{x}(t) =& a_0 - \frac{8a_0}{T}t + \frac{14a_0}{T^2}t^2 - \frac{28t}{T^2}v_0 - \frac{12t}{T^2}v_f \\ &- \frac{20a_0t^3}{3T^3} - \frac{40p_0}{T^3}t + \frac{40p_f}{T^3}t + \frac{64v_0}{T^3}t^2 \\ &+ \frac{36v_f}{T^3}t^2 + \frac{100p_0}{T^4}t^2 - \frac{100p_f}{T^4}t^2 - \frac{100t^3v_0}{3T^4} \\ &- \frac{20v_f}{T^4}t^3 - \frac{160p_0t^3}{3T^5} + \frac{160p_ft^3}{3T^5}.\end{aligned} \tag{80}$$

Introducing again the variable $\xi = t/T \in [0,1]$ and letting $T \to \infty$, yields

$$\lim_{T\to\infty} \ddot{x}(\xi t) = -\frac{20a_0}{3}\xi^3 + 14a_0\xi^2 - 8a_0\xi + a_0 \tag{81}$$

For $a_0 = 0$, the right hand side is trivially zero. For $a_0 \neq 0$, the above can be refactored, and for $\xi \in [0,1]$:

$$\lim_{T\to\infty} \frac{\ddot{x}(\xi T)}{a_0} = -\frac{20}{3}\xi^3 + 14\xi^2 - 8\xi + 1 \in \left[-\frac{29}{75}, 1\right] \tag{82}$$

From this follow (59)-(60).

2) Fixed position: The acceleration trajectory in this case is

$$\begin{aligned}\ddot{x}(t) =& a_0 - \frac{5a_0}{T}t + \frac{5a_0}{T^2}t^2 - \frac{10t}{T^2}v_0 - \frac{5a_0t^3}{3T^3} \\ &- \frac{10p_0}{T^3}t + \frac{10p_f}{T^3}t + \frac{10v_0}{T^3}t^2 + \frac{10p_0}{T^4}t^2 \\ &- \frac{10p_f}{T^4}t^2 - \frac{10t^3v_0}{3T^4} - \frac{10p_0t^3}{3T^5} + \frac{10p_ft^3}{3T^5}.\end{aligned} \tag{83}$$

Introducing again the variable $\xi = t/T \in [0,1]$ and letting $T \to \infty$, yields

$$\lim_{T\to\infty} \ddot{x}(\xi T) = -\frac{5a_0}{3}\xi^3 + 5a_0\xi^2 - 5a_0\xi + a_0 \tag{84}$$

For $a_0 = 0$, the right hand side is trivially zero. For $a_0 \neq 0$, the above can be refactored, and for $\xi \in [0,1]$:

$$\lim_{T\to\infty} \frac{\ddot{x}(\xi T)}{a_0} = -\frac{5}{3}\xi^3 + 5\xi^2 - 5\xi + 1 \in \left[-\frac{2}{3}, 1\right]. \tag{85}$$

From this follow (59)-(60).

3) Fixed velocity: The acceleration trajectory in this case is

$$\begin{aligned}\ddot{x}(t) =& a_0 - \frac{3a_0}{T}t + \frac{3a_0t^2}{2T^2} - \frac{3t}{T^2}v_0 \\ &+ \frac{3t}{T^2}v_f + \frac{3t^2v_0}{2T^3} - \frac{3t^2v_f}{2T^3}.\end{aligned} \tag{86}$$

Introducing again the variable $\xi = t/T \in [0,1]$ and letting $T \to \infty$, yields

$$\lim_{T\to\infty} \ddot{x}(\xi T) = \frac{3a_0}{2}\xi^2 - 3a_0\xi + a_0. \tag{87}$$

For $a_0 = 0$, the right hand side is trivially zero. For $a_0 \neq 0$, the above can be refactored, and for $\xi \in [0,1]$:

$$\lim_{T\to\infty} \frac{\ddot{x}(\xi T)}{a_0} = \frac{3}{2}\xi^2 - 3\xi + 1 \in \left[-\frac{1}{2}, 1\right]. \tag{88}$$

From this follow (59)-(60).

ACKNOWLEDGEMENT

The Flying Machine Arena is the result of contributions of many people, a full list of which can be found at `http://flyingmachinearena.org/`.

This research was supported by the Swiss National Science Foundation (SNSF).

REFERENCES

[1] S. Bellens, J. De Schutter, and H. Bruyninckx, "A hybrid pose/wrench control framework for quadrotor helicopters," in *IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2012, pp. 2269–2274.

[2] R. Ritz, M. W. Mueller, M. Hehn, and R. D'Andrea, "Cooperative quadrocopter ball throwing and catching," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2012, pp. 4972–4978.

[3] G. M. Hoffmann, H. Huang, S. L. Waslander, and C. J. Tomlin, "Quadrotor helicopter flight dynamics and control: Theory and experiment," in *AIAA Guidance, Navigation, and Control Conference*, 2007, pp. 1–20.

[4] S. Grzonka, G. Grisetti, and W. Burgard, "A fully autonomous indoor quadrotor," *IEEE Transactions on Robotics*, vol. 28, no. 1, pp. 90–100, 2012.

[5] C. Richter, A. Bry, and N. Roy, "Polynomial trajectory planning for quadrotor flight," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2013.

[6] G. M. Hoffmann, S. L. Waslander, and C. J. Tomlin, "Quadrotor helicopter trajectory tracking control," in *AIAA Guidance, Navigation and Control Conference and Exhibit*, Honolulu, Hawaii, USA, August 2008, pp. 1–14.

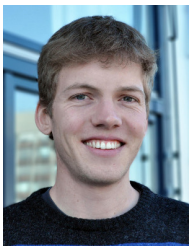
[7] I. D. Cowling, O. A. Yakimenko, J. F. Whidborne, and A. K. Cooke, "A prototype of an autonomous controller for a quadrotor UAV," in *European Control Conference (ECC)*, Kos, Greece, 2007, pp. 1–8.

[8] Y. Bouktir, M. Haddad, and T. Chettibi, "Trajectory planning for a quadrotor helicopter," in *Mediterranean Conference on Control and Automation*, Jun. 2008, pp. 1258–1263.

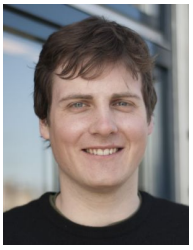
[9] D. Mellinger and V. Kumar, "Minimum snap trajectory generation and control for quadrotors," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2011, pp. 2520–2525.

[10] M. Hehn and R. D'Andrea, "Quadrocopter trajectory generation and control," in *IFAC World Congress*, vol. 18, no. 1, 2011, pp. 1485–1491.

[11] M. P. Vitus, W. Zhang, and C. J. Tomlin, "A hierarchical method for stochastic motion planning in uncertain environments," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2012, pp. 2263–2268.
[12] I. D. Cowling, O. A. Yakimenko, J. F. Whidborne, and A. K. Cooke, "Direct method based control system for an autonomous quadrotor," *Journal of Intelligent & Robotic Systems*, vol. 60, no. 2, pp. 285–316, 2010.
[13] J. Wang, T. Bierling, L. Höcht, F. Holzapfel, S. Klose, and A. Knoll, "Novel dynamic inversion architecture design for quadrocopter control," in *Advances in Aerospace Guidance, Navigation and Control*. Springer, 2011, pp. 261–272.
[14] M. W. Achtelik, S. Lynen, M. Chli, and R. Siegwart, "Inversion based direct position control and trajectory following for micro aerial vehicles," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2013, pp. 2933–2939.
[15] M. Fliess, J. Lévine, P. Martin, and P. Rouchon, "Flatness and defect of non-linear systems: introductory theory and examples," *International journal of control*, vol. 61, no. 6, pp. 1327–1361, 1995.
[16] R. M. Murray, M. Rathinam, and W. Sluis, "Differential flatness of mechanical control systems: A catalog of prototype systems," in *ASME International Mechanical Engineering Congress and Exposition*, 1995.
[17] N. Faiz, S. Agrawal, and R. Murray, "Differentially flat systems with inequality constraints: An approach to real-time feasible trajectory generation," *Journal of Guidance, Control, and Dynamics*, vol. 24, no. 2, pp. 219–227, 2001.
[18] C. Louembet, F. Cazaurang, and A. Zolghadri, "Motion planning for flat systems using positive b-splines: An lmi approach," *Automatica*, vol. 46, no. 8, pp. 1305–1309, 2010.
[19] M. W. Mueller, *Quadrocopter trajectory generator*, Zurich, Switzerland, 2015. [Online]. Available: https://github.com/markwmuller/RapidQuadrocopterTrajectories
[20] M. W. Mueller and R. D'Andrea, "A model predictive controller for quadrocopter state interception," in *European Control Conference*, 2013, pp. 1383–1389.
[21] M. W. Mueller, M. Hehn, and R. D'Andrea, "A computationally efficient algorithm for state-to-state quadrocopter trajectory generation and feasibility verification," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2013, pp. 3480–3486.
[22] P. Martin and E. Salaün, "The true role of accelerometer feedback in quadrotor control," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2010, pp. 1623–1629.
[23] R. Mahony, V. Kumar, and P. Corke, "Aerial vehicles: Modeling, estimation, and control of quadrotor," *IEEE robotics & automation magazine*, vol. 19, no. 3, pp. 20–32, 2012.
[24] P. H. Zipfel, *Modeling and Simulation of Aerospace Vehicle Dynamics Second Edition*. AIAA, 2007.
[25] D. P. Bertsekas, *Dynamic Programming and Optimal Control, Vol. I*. Athena Scientific, 2005.
[26] P. Borwein and T. Erdélyi, *Polynomials and Polynomial Inequalities*, ser. Graduate Texts in Mathematics Series. Springer, 1995.
[27] M. Achtelik, S. Weiss, M. Chli, and R. Siegwart, "Path planning for motion dependent state estimation on micro aerial vehicles," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2013, pp. 3926–3932.
[28] M. Hehn, R. Ritz, and R. D'Andrea, "Performance benchmarking of quadrotor systems using time-optimal control," *Autonomous Robots*, vol. 33, pp. 69–88, 2012.
[29] L. Meier, P. Tanskanen, L. Heng, G. Lee, F. Fraundorfer, and M. Pollefeys, "PIXHAWK: A micro aerial vehicle design for autonomous flight using onboard computer vision," *Autonomous Robots*, vol. 33, no. 1-2, pp. 21–39, 2012.
[30] S. Lupashin, M. Hehn, M. W. Mueller, A. P. Schoellig, M. Sherback, and R. D'Andrea, "A platform for aerial robotics research and demonstration: The Flying Machine Arena," *Mechatronics*, vol. 24, no. 1, pp. 41 – 54, 2014.
[31] D. Gurdan, J. Stumpf, M. Achtelik, K.-M. Doth, G. Hirzinger, and D. Rus, "Energy-efficient autonomous four-rotor flying robot controlled at 1 kHz," in *IEEE International Conference on Robotics and Automation (ICRA)*, April 2007, pp. 361–366.
[32] M. W. Mueller, S. Lupashin, and R. D'Andrea, "Quadrocopter ball juggling," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2011, pp. 5113–5120.
[33] C. E. García, D. M. Prett, and M. Morari, "Model predictive control: Theory and practice–a survey," *Automatica*, vol. 25, no. 3, pp. 335 – 348, 1989.
[34] E. Frazzoli, M. Dahleh, and E. Feron, "Real-time motion planning for agile autonomous vehicles," *Journal of Guidance Control and Dynamics*, vol. 25, no. 1, pp. 116–129, 2002.
[35] M. Sherback, O. Purwin, and R. D'Andrea, "Real-time motion planning and control in the 2005 cornell robocup system," in *Robot Motion and Control*, ser. Lecture Notes in Control and Information Sciences, K. Kozlowski, Ed. Springer London, 2006, vol. 335, pp. 245–263.



Mark W. Mueller is currently a doctoral candidate at the Institute for Dynamic Systems and Control at the ETH Zurich. He received the B.Eng. degree in Mechanical Engineering from the University of Pretoria in 2009, and the M.Sc. degree in Mechanical Engineering from the ETH Zurich in 2011. He received awards for the best mechanical engineering thesis, and the best aeronautical thesis, for his bachelors thesis in 2008, and received the Jakob Ackeret award from the Swiss Association of Aeronautical Sciences for his masters thesis in 2011. His masters studies were supported by a scholarship from the Swiss Government.



Markus Hehn received the Diplom-Ingenieur degree in Mechatronics from TU Darmstadt in 2009, and the Doctor of Sciences degree from ETH Zurich in 2014. He won a scholarship from Robert Bosch GmbH for his graduate studies, and the Jakob-Ackeret-prize of the Swiss Association of Aeronautical Sciences for his doctoral thesis. He has worked on optimizing the operating strategy of Diesel engines to reduce emissions and fuel consumption, on the characterization of component load profiles for axle split hybrid vehicle drivetrains, and on the performance development of Formula One racing engines. His main research interest are the control and trajectory generation for multi-rotor vehicles during fast maneuvering, optimality-based maneuvers, multi-vehicle coordination, and learning algorithms.

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. While on leave from Cornell, from 2003 to 2007, he co-founded Kiva Systems, where he led the systems architecture, robot design, robot navigation and coordination, and control algorithms efforts. He is currently professor of Dynamic Systems and Control at ETH Zurich.