Computationally Efficient Trajectory Generation for Fully-Actuated Multirotor Vehicles

Dario Brescianini, *Student Member, IEEE*, and Raffaello D'Andrea, *Fellow, IEEE*

***Abstract*—This paper presents a computationally efficient method of generating state-to-state trajectories for fully-actuated multirotor vehicles. The approach consists of computing translational and rotational motion primitives that guide the vehicle from any initial state, defined by position, velocity and attitude, to any end state in a given time, and subsequently verifying the motion primitives' feasibility. Computationally light-weight motion primitives for which closed-form solutions exist are presented and an efficient method to test their feasibility is derived. The algorithm is shown to be able to generate trajectories and verify their feasibility within a few microseconds and can thus be used as an implicit feedback law or in high level path planners that involve evaluating a large number of possible trajectories to achieve some high level goal. The algorithm's performance is analyzed by comparing it with time-optimal trajectories. An experimental demonstration that requires the computation of trajectories for a large set of end states in real time is used to evaluate the approach.**

***Index Terms*—Trajectory generation, motion control, aerial robotics.**

I. Introduction

MULTIROTOR vehicles have become very popular aerial robotic platforms due to their high maneuverability and ability to hover. However, one of the limitations of traditional multirotor vehicles is their inherent under-actuation, i.e. their inability to independently control their thrust and torque in all three dimensions. In order to increase performance criteria such as flight duration or payload, all rotors are typically arranged in a single plane, thereby limiting the thrust to a single direction and coupling the vehicle's translational and rotational dynamics. This limits not only the set of feasible trajectories, but also the vehicle's ability to instantaneously resist arbitrary force and torque disturbances as required when flying high precision maneuvers or when physically interacting with the environment. To overcome these limitations, several novel multirotor vehicle designs with non-planar rotor configurations have been developed in the past years such as the hexrotor vehicles [1]–[5] or the octorotor vehicle [6]. These multirotor vehicles are capable of independently generating thrust and torque in any direction and hence allow control of all of their six degrees-of-freedom independently.

A. Goal and Motivation

A key feature required to exploit the full dynamic capabilities of these novel multirotor vehicles is a trajectory generator that can compute a large set of position and attitude flight paths in real time while respecting the system dynamics and input constraints. In many of the envisioned application areas such as aerial manipulation or filming, the environment may constantly change: the target object may move; large disturbances may throw the vehicle off its originally planned course; or information about the environment may only become available in mid-flight. In such dynamic environments, pre-planned trajectories may become suboptimal or even infeasible, and thus a method to constantly adapt the trajectory in real time is required. Furthermore, in many of these scenarios there exist multiple trajectories that achieve the same high level goal, and hence, a trajectory generator is needed that is capable of rapidly evaluating a large set of trajectories and selecting the best trajectory with respect to some performance measure.

The authors are members of the Institute for Dynamic Systems and Control, ETH Zurich, Switzerland. {bdario, rdandrea}@ethz.ch

B. Related Work

A number of trajectory generation algorithms for traditional multirotor vehicles have been presented in recent years. These algorithms can roughly be divided into three groups: A first group of algorithms generates trajectories based on path primitives, for example lines [7], polynomials [8], or splines [9], and subsequently parametrizes the paths in time such that the dynamic constraints are satisfied. A second group consists of algorithms that make use of the differential flatness of the system to approximate the input constraints by constraints on position derivatives. Trajectories are then generated by solving an optimal control problem on the translational kinematics with the objective of, for example, minimizing the maneuver duration [10] or some position derivatives [11], [12]. A computationally very inexpensive method of this group is presented in [13], where the input constraints are neglected when solving the optimization problem and an efficient method is used subsequently to check whether the resulting trajectories violate the input constraints. Finally, a third group of algorithms generates trajectories by solving an optimal control problem on the full system dynamics numerically, either leveraging Pontryagin's minimum principle [14] or using numerical optimal control [15], [16].

Although the aforementioned algorithms can be used to generate trajectories for the novel fully-actuated multirotor vehicles, they do not take full advantage of the dynamic capabilities of these vehicles as they only generate attitude trajectories that are directly coupled with the vehicle's position trajectory. Some algorithms to generate decoupled position and attitude trajectories have been developed for spacecraft applications. For example in [17] and [18], trajectories are generated using separate position and attitude path primitives that are then parametrized in time to ensure their feasibility with respect to the input constraints. However, these algorithms are

usually not fast enough to apply in real time to multirotor vehicles whose position and attitude control loops typically run at rates of 50-100Hz [19], [20] and hence demand the trajectories to be generated in a few milliseconds.

C. Contribution

This paper presents and analyzes a trajectory generation algorithm for fully-actuated multirotor vehicles capable of rapidly generating thousands of trajectories which guide the vehicle from any initial state to any end state in a given time. The algorithm is similar to the approach presented in [13] and is based on concatenating computationally light-weight motion primitives for which closed-form solutions exist. When generating the motion primitives, the input constraints are neglected and their feasibility is then validated a posteriori by relating the motion primitives' position and attitude trajectories to the input constraints using the multirotor vehicle's system dynamics. In [13], only traditional multirotor vehicles with planar rotor configurations were considered, i.e. vehicles that can only produce a thrust in a single direction. For these aerial vehicles, it is sufficient to only compute translational motion primitives as this fully defines the vehicle's state and control inputs (up to a yaw rotation). However, this is not sufficient for fully-actuated aerial vehicles and this paper thus presents a method to generate both translational and rotational motion primitives and introduces sufficient conditions to verify their feasibility.

When computing the motion primitives, the vehicle's ability to control all of its six degrees of freedom independently is used and separate motion primitives for the translational and rotational motion are computed, each by solving an optimal control problem. For the translational motion primitive, the objective is to minimize the jerk, and for the rotational motion primitive, the objective is to minimize the trajectory's rotation vector acceleration, an approximation of the angular acceleration. In both cases, the solution trajectories are characterized by polynomials in time, a key property that is used for the fast and efficient validation of the trajectories' feasibility.

Since the algorithm is able to generate trajectories for arbitrary initial states, final states and maneuver duration, it can be applied to a large class of trajectory generation problems. Furthermore, the algorithm is shown to be computationally efficient and can generate approximately 500 000 trajectories per second when implemented on a standard laptop computer. Due to these two properties, the algorithm is well suited to be used in high level path planners such as probabilistic roadmap [21] or rapidly exploring random tree algorithms [22] that involve evaluating large numbers of candidate trajectories, or as an implicit feedback law similar to model predictive control [23].

The algorithm's performance is assessed through an experimental demonstration that requires doing a search over a large set of possible end states in real time. The goal is for a fully-actuated multirotor vehicle to catch a thrown ball in a small pouch. The trajectory generator is embedded in a high level path planner that evaluates a set of possible end states such that the ball is caught in the pouch. Because the ball's flight path varies for each throw, the trajectories cannot be pre-planned and have to be generated in real time.

D. Outline

The remainder of this paper is organized as follows: Section II introduces preliminaries on two attitude representations that are used throughout the paper. In Section III, the system dynamics and the input constraints of the multirotor vehicles considered in this paper are presented. In Section IV, the trajectory generation problem is formally stated. In Section V, motion primitives that guide the vehicle from any initial to any final state are introduced, and a method to determine their feasibility is presented in Section VI. Section VII presents a trajectory generation algorithm that fulfills the trajectory generation problem. The algorithm's performance compared to the multirotor vehicle's physical limits as well as results on the computational costs are presented in Section VIII. An experimental evaluation of the trajectory generation algorithm is presented in Section IX, and the paper is concluded in Section X.

II. Preliminaries

In this section, two attitude representations that will be used throughout the paper are introduced: the rotation vector $\boldsymbol{r} \in \mathbb{R}^3$ and the rotation matrix $\boldsymbol{R} \in \text{SO}(3)$, where $\text{SO}(3)$ denotes the Special Orthogonal Group, defined as

$$\begin{aligned}\text{SO}(3) = \{\boldsymbol{R} \in \mathbb{R}^{3\times 3} | \boldsymbol{R}^T\boldsymbol{R} = \boldsymbol{R}\boldsymbol{R}^T = \boldsymbol{I} \\ \text{and } \det(\boldsymbol{R}) = 1\},\end{aligned} \tag{1}$$

where $\boldsymbol{I} \in \mathbb{R}^{3\times 3}$ is the identity matrix. The rotation matrix representation is convenient for rotating vectors or expressing the attitude kinematics. However, the constraints (1) on the group of rotation matrices renders it difficult to use them for attitude trajectory generation. Rotation vectors on the contrary are an unconstrained three-parameter representation, but rotating vectors or computing the attitude kinematics is more complicated. For these reasons, the attitude trajectories in this paper will be planned using rotation vectors, and rotation matrices will be used to represent attitudes in all other cases. In the following, the conversion between the two representations and their relation to angular velocity is presented.

Any arbitrary rotation can be described by a rotation axis and a rotation angle about this axis (see for example [24], and references therein). Let $\boldsymbol{n} \in \mathbb{S}^2$ be a unit vector which represents the axis of rotation, where $\mathbb{S}^2 = \{\boldsymbol{y} \in \mathbb{R}^3 \mid \boldsymbol{y}^T\boldsymbol{y} = 1\}$, and let $\varphi \in \mathbb{R}$ be the rotation angle. The rotation vector $\boldsymbol{r}$ describing this rotation is defined to be

$$\boldsymbol{r} = \varphi\boldsymbol{n}. \tag{2}$$

Note that the rotation vector representation is not unique. In fact, any rotation vector $\boldsymbol{r} = \varphi\boldsymbol{n} + 2k\pi\boldsymbol{n}$, with k any integer, represents the same physical attitude. The rotation matrix $\boldsymbol{R}$ corresponding to the physical attitude represented by $\boldsymbol{r}$ is given by [24]

$$\boldsymbol{R} = e^{[\boldsymbol{r}]}, \tag{3}$$

$$= \boldsymbol{I} + \frac{\sin\|\boldsymbol{r}\|}{\|\boldsymbol{r}\|}[\boldsymbol{r}] + \frac{1-\cos\|\boldsymbol{r}\|}{\|\boldsymbol{r}\|^2}[\boldsymbol{r}]^2, \tag{4}$$

where $\|\cdot\|$ denotes the Euclidean norm $\sqrt{\boldsymbol{r}^T\boldsymbol{r}}$ and $[\boldsymbol{r}]$ is the skew-symmetric cross-product matrix representation of $\boldsymbol{r}$,

$$[\boldsymbol{r}] = \begin{bmatrix} 0 & -r_3 & r_2 \\ r_3 & 0 & -r_1 \\ -r_2 & r_1 & 0 \end{bmatrix}. \tag{5}$$

Conversely, given a rotation matrix $\boldsymbol{R}$, the rotation vector is obtained by

$$[\boldsymbol{r}] = \log \boldsymbol{R}, \tag{6}$$

$$= \frac{\varphi}{2\sin\varphi}\left(\boldsymbol{R} - \boldsymbol{R}^T\right), \tag{7}$$

where φ satisfies $1 + 2\cos\varphi = \operatorname{trace}(\boldsymbol{R})$ and $\varphi \in [0, \pi]$.

Let $\boldsymbol{R}(t)$ be a curve in $\mathrm{SO}(3)$ describing the attitude of a rigid body relative to a fixed coordinate frame. The relation between the angular velocity in body-fixed coordinates $\boldsymbol{\omega}$ and the rotation matrix and its temporal derivative is given by

$$[\boldsymbol{\omega}] = \boldsymbol{R}^T\dot{\boldsymbol{R}}. \tag{8}$$

An analogous relation can be obtained when the attitude of a body is described by a rotation vector $\boldsymbol{r}(t)$. Differentiating (4) with respect to time and inserting it into (8) yields [24]

$$\boldsymbol{\omega} = \boldsymbol{W}(\boldsymbol{r})\dot{\boldsymbol{r}}, \tag{9}$$

where $\boldsymbol{W}(\boldsymbol{r}) = \boldsymbol{I}$ if $\|\boldsymbol{r}\| = 0$, and otherwise

$$\boldsymbol{W}(\boldsymbol{r}) = \boldsymbol{I} - \frac{1-\cos\|\boldsymbol{r}\|}{\|\boldsymbol{r}\|^2}[\boldsymbol{r}] + \frac{\|\boldsymbol{r}\| - \sin\|\boldsymbol{r}\|}{\|\boldsymbol{r}\|^3}[\boldsymbol{r}]^2. \tag{10}$$

III. Dynamic Model

In this section, the multirotor vehicle dynamics and its input constraints are presented. Only fully-actuated multirotor vehicles are considered, i.e. multirotor vehicles that can generate thrust and torque in all three dimensions independently of each other. For ease of notation, vectors may be expressed as n-tuples $\boldsymbol{x} = (x_1, x_2, \ldots, x_n)$ with dimensions and stacking clear from context.

A. Equations of Motion

The multirotor vehicle is modelled as a rigid body. The translational degrees-of-freedom are described by the position of the multirotor vehicle's center of mass $\boldsymbol{x} = (x_1, x_2, x_3)$, expressed in an inertial frame, and its rotational degrees-of-freedom are parametrized using the rotation matrix $\boldsymbol{R} \in \mathrm{SO}(3)$.

The multirotor vehicle's control inputs are considered to be the mass-normalized collective thrust $\boldsymbol{f} = (f_1, f_2, f_3)$ and the angular velocity $\boldsymbol{\omega} = (\omega_1, \omega_2, \omega_3)$, both expressed in the vehicle's body-fixed coordinate frame as depicted in Fig. 1. The thrust dynamics of multirotor vehicles are typically much faster than their rigid body dynamics, and hence it is assumed that the commanded thrust can be achieved instantaneously. Likewise, it is assumed that the commanded angular velocity can be changed instantaneously. Because of multirotor vehicles' low rotational inertia and their ability to produce large torques due to the off-center mounting of their rotors, it is assumed that a high-bandwidth on-board controller can track the angular velocity commands sufficiently fast using gyroscope feedback. The vehicle's attitude dynamics are thus neglected and the equations of motion can be written as

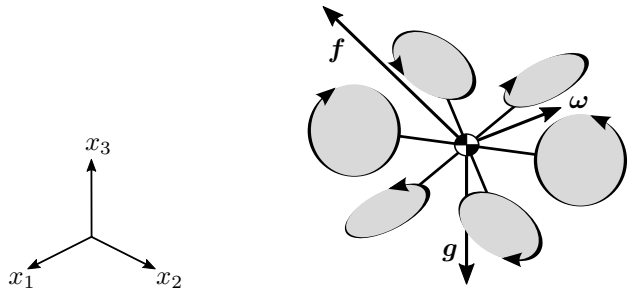


Fig. 1: Illustration of a fully-actuated multirotor vehicle with its position described by $\boldsymbol{x} = (x_1, x_2, x_3)$ in the inertial frame. The vehicle's control inputs are the mass-normalized collective thrust $\boldsymbol{f}$ and the angular velocity $\boldsymbol{\omega}$. In addition, gravity $\boldsymbol{g}$ is acting upon the vehicle.

$$\ddot{\boldsymbol{x}} = \boldsymbol{R}\boldsymbol{f} - \boldsymbol{g}, \tag{11}$$

$$\dot{\boldsymbol{R}} = \boldsymbol{R}[\boldsymbol{\omega}], \tag{12}$$

where $\boldsymbol{g}$ is the acceleration due to gravity, expressed in the inertial frame.

Note that more detailed models for the dynamics of multirotor vehicles exist that incorporate, for example, aerodynamic effects such as drag [25]. However, the preceding model captures the most relevant dynamics and greatly simplifies the trajectory generation problem, which then again allows compensation for model inaccuracies by continuously re-planning the trajectories. It should further be noted that by taking the control inputs as the collective thrust and angular velocity, the vehicle's rotor configuration, i.e. the number of rotors, their positions and orientations, is hidden from the equations of motion by the on-board angular velocity controller that computes the individual rotor thrusts based on the angular velocity error and the commanded collective thrust. As a result, the differential equations (11) and (12) describing the system dynamics hold true for any fully-actuated multirotor vehicle, such as the vehicles presented in [1]–[6].

B. Input Constraints

It is assumed that the control inputs are subject to saturations. The attainable collective thrust is constrained to the polyhedron

$$\boldsymbol{A}_f\boldsymbol{f} \preceq \boldsymbol{b}_f, \tag{13}$$

where the symbol $\preceq$ denotes componentwise inequality. The faces of the polyhedron described by (13) encode the thrust limits in different directions that are due to the orientation and saturation limits of the individual rotors. Examples of attainable thrust volumes for fully-actuated multirotor vehicles can be found in [4] and [5].

The angular velocity is assumed to be limited to the polyhedron

$$\boldsymbol{A}_\omega\boldsymbol{\omega} \preceq \boldsymbol{b}_\omega, \tag{14}$$

where the limits can be due to, for example, the measurement range of the gyroscope used for feedback or the range for which the angular velocity controller responds sufficiently fast to angular velocity commands such that the simplified system dynamics (11) and (12) describe the system behaviour well. An example of angular velocity constraints for the multirotor vehicle [6] is given in Appendix A.

Without loss of generality, it is assumed in the following that all rows of $\boldsymbol{A}_f$ and $\boldsymbol{A}_\omega$ are unit vectors.

IV. Problem Statement

The trajectory generation problem addressed in this paper can be formulated as follows: given an initial state at time $t = 0$, consisting of position, velocity and attitude, find control inputs $\boldsymbol{f}(t)$ and $\boldsymbol{\omega}(t)$, $t \in [0, T]$, that steer the multirotor vehicle to a desired final state at time $t = T$, while satisfying the system dynamics (11) and (12) and input constraints (13) and (14). Furthermore, the generation of trajectories should be computationally inexpensive such that a large number of trajectories can be computed in real time.

The approach presented in the following consists of two steps: In a first step, motion primitives guiding the vehicle from any initial to any desired end state in a given time are planned while the input constraints are ignored. In a second step, the control inputs are recovered from the motion primitives' position and attitude trajectory using the system dynamics and then verified for input feasibility. If feasibility cannot be established, the two steps are recursively performed on subintervals and the resulting motion primitives are concatenated.

V. Motion Primitive Generation

In this section, motion primitives that guide the multirotor vehicle from any initial state to any desired end state in a given time are presented. As in [13] for traditional multirotor vehicles, the motion primitives are characterized by polynomials in time. However, since fully-actuated multirotor vehicles can independently control their position and attitude, it is not sufficient to only compute a translational motion primitive as for traditional multirotor vehicles, but a rotational motion primitive also needs to be computed.

In the following, separate motion primitives for the vehicle's position and attitude are planned in the position coordinates $\boldsymbol{x}$ and attitude coordinates $\boldsymbol{r}$, respectively. It can be seen from the system dynamics (11) and (12) that in order to be able to recover the motion primitives' corresponding control inputs $\boldsymbol{f}(t)$ and $\boldsymbol{\omega}(t)$, the position trajectory $\boldsymbol{x}(t)$ needs to be at least twice differentiable with respect to time, and the attitude trajectory $\boldsymbol{r}(t)$, or equivalently $\boldsymbol{R}(t)$, at least once. Without loss of generality due to time invariance, the motion primitives are planned on the interval $[0, T]$.

A. Translational Motion Primitive

The goal of the translational motion primitive is to guide the vehicle from any initial position and velocity to any desired end position and velocity in time T. We seek to find the motion primitive that minimizes the average jerk squared on the time interval $[0, T]$, i.e.

$$J_{\text{trans}} = \frac{1}{T}\int_0^T \|\dddot{\boldsymbol{x}}(t)\|^2 dt. \tag{15}$$

This cost function is chosen because it is computationally convenient, a closed-form solution exists, and it works well in practice. Furthermore, it yields a position trajectory that is three times differentiable with respect to time. Consequently, the corresponding control input $\boldsymbol{f}$ can be made continuous even when concatenating multiple motion primitives and is therefore easy to track.

The optimal jerk $\dddot{\boldsymbol{x}}(t)$ minimizing (15) can be computed using Pontryagin's minimum principle (see for example [26]) and is derived in Appendix B. Its corresponding position, velocity and acceleration trajectories can be shown to be of the form

$$\boldsymbol{x}(t) = \tfrac{\boldsymbol{c}_1}{120}t^5 + \tfrac{\boldsymbol{c}_2}{24}t^4 + \tfrac{\boldsymbol{c}_3}{6}t^3 + \tfrac{\boldsymbol{c}_4}{2}t^2 + \boldsymbol{c}_5 t + \boldsymbol{c}_6, \tag{16}$$

$$\dot{\boldsymbol{x}}(t) = \tfrac{\boldsymbol{c}_1}{24}t^4 + \tfrac{\boldsymbol{c}_2}{6}t^3 + \tfrac{\boldsymbol{c}_3}{2}t^2 + \boldsymbol{c}_4 t + \boldsymbol{c}_5, \tag{17}$$

$$\ddot{\boldsymbol{x}}(t) = \tfrac{\boldsymbol{c}_1}{6}t^3 + \tfrac{\boldsymbol{c}_2}{2}t^2 + \boldsymbol{c}_3 t + \boldsymbol{c}_4. \tag{18}$$

The constraints on the initial and final position and velocity of the motion primitive only partially define the trajectories (16)-(18). The motion primitive's initial and final acceleration can either be left free and subject to optimization, or they can be used to ensure smooth transitions on the control input $\boldsymbol{f}$ when concatenating multiple motion primitives by setting the initial acceleration to be equal to the final acceleration of the preceding motion primitive, and likewise for the final acceleration. If not mentioned otherwise, it will be assumed in the remainder of this paper that also the initial and final acceleration are defined, such that $(\boldsymbol{x}(0), \dot{\boldsymbol{x}}(0), \ddot{\boldsymbol{x}}(0)) = (\boldsymbol{p}_0, \boldsymbol{v}_0, \boldsymbol{a}_0)$ and $(\boldsymbol{x}(T), \dot{\boldsymbol{x}}(T), \ddot{\boldsymbol{x}}(T)) = (\boldsymbol{p}_T, \boldsymbol{v}_T, \boldsymbol{a}_T)$. The components of the coefficients $\boldsymbol{c}_1, \ldots, \boldsymbol{c}_6$ along the i-th axis are then given by

$$\begin{bmatrix} c_{1,i} \\ c_{2,i} \\ c_{3,i} \end{bmatrix} = \frac{1}{T^5}\begin{bmatrix} 720 & -360T & 60T^2 \\ -360T & 168T^2 & -24T^3 \\ 60T^2 & -24T^3 & 3T^4 \end{bmatrix}\begin{bmatrix} \Delta p_i \\ \Delta v_i \\ \Delta a_i \end{bmatrix}, \tag{19}$$

and

$$\begin{bmatrix} c_{4,i} \\ c_{5,i} \\ c_{6,i} \end{bmatrix} = \begin{bmatrix} a_{0,i} \\ v_{0,i} \\ p_{0,i} \end{bmatrix}, \tag{20}$$

where, for example, $c_{1,i}$ is the component of $\boldsymbol{c}_1$ along the i-th axis and

$$\begin{bmatrix} \Delta p_i \\ \Delta v_i \\ \Delta a_i \end{bmatrix} = \begin{bmatrix} p_{T,i} - p_{0,i} - v_{0,i}T - \frac{1}{2}a_{0,i}T^2 \\ v_{T,i} - v_{0,i} - a_{0,i}T \\ a_{T,i} - a_{0,i} \end{bmatrix}. \tag{21}$$

The cost of the motion primitive can then be computed to be

$$\begin{aligned} J_{\text{trans}} = {} & \frac{1}{20}\boldsymbol{c}_1^T\boldsymbol{c}_1 T^4 + \frac{1}{4}\boldsymbol{c}_1^T\boldsymbol{c}_2 T^3 + \frac{1}{3}\left(\boldsymbol{c}_1^T\boldsymbol{c}_3 + \boldsymbol{c}_2^T\boldsymbol{c}_2\right)T^2 \\ & + \boldsymbol{c}_2^T\boldsymbol{c}_3 T + \boldsymbol{c}_3^T\boldsymbol{c}_3. \end{aligned} \tag{22}$$

B. Rotational Motion Primitive

The goal of the rotational motion primitive is to guide the vehicle from any initial attitude to any desired end attitude in time T. We seek to find the motion primitive that minimizes the average rotation vector acceleration squared on the time interval $[0, T]$, i.e.

$$J_{\text{rot}} = \frac{1}{T}\int_0^T \|\ddot{\boldsymbol{r}}(t)\|^2 dt. \tag{23}$$

Similar to the translational motion primitive, this cost function is chosen because it is computationally convenient, has a closed-form solution, works well in practice, and yields an attitude trajectory that is twice differentiable with respect to time. Consequently, the corresponding control input $\boldsymbol{\omega}$ can be made continuous and is thus easy to track even when concatenating multiple motion primitives, which is exploited in Section VII. In [27], it is shown that the rotation vector acceleration $\ddot{\boldsymbol{r}}$ approaches the angular acceleration $\dot{\boldsymbol{\omega}}$ if either

- the rotation is small, i.e. $\|\boldsymbol{r}(t)\| \to 0$,
- the rotation is slow, i.e. $\|\dot{\boldsymbol{r}}(t)\| \to 0$,
- or the rotation axis does not vary considerably, i.e. $\angle(\boldsymbol{r}, \dot{\boldsymbol{r}}) \to 0$ and $\angle(\boldsymbol{r}, \ddot{\boldsymbol{r}}) \to 0$.

Minimizing the cost (23) can therefore be interpreted as an approximation of minimizing the average angular acceleration squared,

$$J = \frac{1}{T}\int_0^T \|\dot{\boldsymbol{\omega}}(t)\|^2 dt, \tag{24}$$

however, in general, the latter does not admit a closed-form solution and is hence not well-suited to generate a large set of trajectories in real time.

Let $\boldsymbol{R}_0$ and $\boldsymbol{R}_T$ be the initial and final attitude, respectively. In order to be able to compute the maximum rotation angle in closed-form (see Section VI-A) and for the solution to approximate the minimum angular acceleration trajectory well, the motion primitive is always planned from an initial attitude of $\boldsymbol{r}(0) = 0$ to a final attitude of $\boldsymbol{r}(T) = \boldsymbol{r}_e$ such that $\|\boldsymbol{r}(t)\|$ remains small, where $\boldsymbol{r}_e$ is the rotation between $\boldsymbol{R}_0$ and $\boldsymbol{R}_T$,

$$[\boldsymbol{r}_e] = \log(\boldsymbol{R}_0^T \boldsymbol{R}_T), \tag{25}$$

and the resulting motion primitive is then rotated by $\boldsymbol{R}_0$ to satisfy the original attitude conditions[1].

As with the translational motion primitive, the optimal rotation vector acceleration $\ddot{\boldsymbol{r}}(t)$ minimizing (23) can be computed using Pontryagin's minimum principle (see Appendix B for a derivation) and yields attitude and angular velocity trajectories of the form

$$\boldsymbol{R}(t) = \boldsymbol{R}_0 e^{[\boldsymbol{r}(t)]}, \tag{26}$$
$$\boldsymbol{\omega}(t) = \boldsymbol{W}(\boldsymbol{r}(t))\dot{\boldsymbol{r}}(t), \tag{27}$$

[1]Note that (25) computes the rotation vector $\boldsymbol{r}_e$ such that $\|\boldsymbol{r}_e\| \leq \pi$, i.e. such that the rotation angle is smaller than π. By adding $2k\pi\boldsymbol{r}_e/\|\boldsymbol{r}_e\|$ to (25), with k being any integer, the motion primitive ends at the same physical attitude but performs an additional k full rotations. Throughout this paper, k is always chosen to be zero.

with

$$\boldsymbol{r}(t) = \tfrac{\boldsymbol{d}_1}{6}t^3 + \tfrac{\boldsymbol{d}_2}{2}t^2 + \boldsymbol{d}_3 t, \tag{28}$$
$$\dot{\boldsymbol{r}}(t) = \tfrac{\boldsymbol{d}_1}{2}t^2 + \boldsymbol{d}_2 t + \boldsymbol{d}_3. \tag{29}$$

Analogous to the translational motion primitive, the constraints on the initial and final attitude of the motion primitive only partially define the attitude and angular velocity trajectory (26)-(29). The initial and final angular velocity can either be left free and subject to optimization, or they can be used to ensure smooth transitions of the control input $\boldsymbol{\omega}$ when concatenating motion primitives by setting the initial angular velocity to be equal to the final angular velocity of the preceding motion primitive, and analogously for the final angular velocity. If not mentioned otherwise, it will be assumed in the remainder of this paper that the initial and final angular velocity $\boldsymbol{\omega}_0$ and $\boldsymbol{\omega}_T$ are also defined. The components of the coefficients $\boldsymbol{d}_1, \boldsymbol{d}_2$ and $\boldsymbol{d}_3$ along the i-th axis are then given by

$$\begin{bmatrix} d_{1,i} \\ d_{2,i} \end{bmatrix} = \frac{1}{T^3}\begin{bmatrix} -12 & 6T \\ 6T & -2T^2 \end{bmatrix}\begin{bmatrix} r_{e,i} - \omega_{0,i}T \\ \tilde{\omega}_{T,i} - \omega_{0,i} \end{bmatrix}, \tag{30}$$

and

$$d_{3,i} = \omega_{0,i}, \tag{31}$$

where $\tilde{\boldsymbol{\omega}}_T$ is defined to be

$$\tilde{\boldsymbol{\omega}}_T := \boldsymbol{W}^{-1}(\boldsymbol{r}_e)\boldsymbol{\omega}_T. \tag{32}$$

The cost of the motion primitive can then be shown to be

$$J_{\text{rot}} = \frac{1}{3}\boldsymbol{d}_1^T\boldsymbol{d}_1 T^2 + \boldsymbol{d}_1^T\boldsymbol{d}_2 T + \boldsymbol{d}_2^T\boldsymbol{d}_2. \tag{33}$$

C. Discussion

Generating the motion primitives can be done very efficiently. In the case of the translational motion primitive, it only requires the evaluation of (19) and (20) for each axis. In the case of the rotational motion primitive, it requires the evaluation of (25) and (32), and subsequently (30) and (31) for each axis. In addition, calculating the costs of the motion primitives is also computationally inexpensive and can be done in closed-form using (22) and (33). If multiple motion primitives achieve the same high level goal, these costs could be used to compare the aggressiveness of the different motion primitives.

VI. Verification of Feasibility

In this section a method to efficiently verify the motion primitives' feasibility is introduced. In [13], the feasibility of the motion primitives was determined by finding extreme points of the motion primitive's position trajectory (and its time derivatives) and relating them to the control input constraints using the differential flatness of traditional multirotor vehicles. Due to the different system dynamics of fully-actuated multirotor vehicles and the decoupled planning of the translational and rotational motion primitive, the feasibility checks of [13] cannot be applied to the motion primitives of Section V and a different method to verify their feasibility is devised.

Although the motion primitives for the position and attitude are planned independently of each other, their feasibility cannot be verified separately as the position and attitude dynamics are coupled by the thrust input. The translational coordinates are expressed in an inertial frame, but the thrust input and its constraints are expressed in the vehicle's body frame and hence depend on the vehicle's attitude. In the following, verifying the motion primitives' feasibility is done by first computing the maximum rotation angle of the rotational motion primitive, and then verifying whether the control inputs satisfy their constraints under any rotation with a rotation angle equal or smaller than the maximum rotation angle.

A. Maximum Rotation Angle

It can be seen from the multirotor vehicle's attitude given in (26) that the maximum rotation angle relative to the vehicle's initial attitude $\boldsymbol{R}_0$ is given by

$$\varphi_{\max} = \max_{0\leq t\leq T} \|\boldsymbol{r}(t)\|, \tag{34}$$

and can be solved by finding the roots of

$$\frac{d}{dt}\left(\boldsymbol{r}^T(t)\boldsymbol{r}(t)\right) = 0. \tag{35}$$

By design of the rotational motion primitive, $\boldsymbol{r}(t)$ always has one root at $t = 0$ and consequently (35) also has one root at $t = 0$. Therefore, finding the other roots of (35) is equivalent to finding the roots of a quartic polynomial, for which closed-form solutions exist [28].

B. Thrust Input Feasibility

The thrust needed during the execution of the motion primitives can be obtained through the translational dynamics and is given by

$$\boldsymbol{f}(t) = \boldsymbol{e}^{[-\boldsymbol{r}(t)]}\boldsymbol{R}_0^T\left(\ddot{\boldsymbol{x}}(t) + \boldsymbol{g}\right). \tag{36}$$

The motion primitives' feasibility with respect to the thrust constraint, i.e.

$$\boldsymbol{A}_f\boldsymbol{f}(t) \preceq \boldsymbol{b}_f, \quad \forall t \in [0, T], \tag{37}$$

is determined by first examining if the initial thrust is feasible,

$$\boldsymbol{A}_f\boldsymbol{R}_0^T\left(\boldsymbol{c}_4 + \boldsymbol{g}\right) \preceq \boldsymbol{b}_f, \tag{38}$$

and afterwards verifying that the thrust remains feasible during the entire motion primitive. This is done by computing a bounding box on the required mass-normalized thrust, expressed in the vehicle's initial body frame $\boldsymbol{R}_0$, and ensuring that all points in the bounding box satisfy the thrust constraints under any rotation with a maximum rotation angle of $\varphi_{\max}$.

Let $\mathcal{H}$ be the bounding box on the mass-normalized thrust rotated into the vehicle's initial body frame, i.e.

$$\mathcal{H} = \{\boldsymbol{y} \in \mathbb{R}^3 | \boldsymbol{h}_{\min} \preceq \boldsymbol{R}_0\boldsymbol{y} \preceq \boldsymbol{h}_{\max}\}, \tag{39}$$

where the lower and upper bounds $\boldsymbol{h}_{\min}$ and $\boldsymbol{h}_{\max}$ are computed such that

$$\boldsymbol{h}_{\min} \preceq \ddot{\boldsymbol{x}}(t) + \boldsymbol{g} \preceq \boldsymbol{h}_{\max}, \quad \forall t \in [0, T]. \tag{40}$$

The lower and upper bounds are expressed in the inertial frame for computational efficiency reasons (see Section VII) and are obtained by evaluating $\ddot{\boldsymbol{x}}(t)$ at the boundaries of the interval $[0, T]$ and by solving for the extrema of the acceleration $\ddot{\boldsymbol{x}}(t)$ along each axis on the interval $[0, T]$, which is essentially a matter of finding the roots of its derivative (a polynomial of order at most two).

The constraint (37) can then be formulated as

$$\boldsymbol{A}_f\boldsymbol{e}^{[-\boldsymbol{r}(t)]}\boldsymbol{h} \preceq \boldsymbol{b}_f, \quad \forall \boldsymbol{h} \in \mathcal{H}, \quad \forall t \in [0, T]. \tag{41}$$

In Appendix C, it is shown that any point $\boldsymbol{y} \in \mathbb{R}^3$ remains in a closed ball B with center $\delta_f\boldsymbol{y}$ and minimal radius $\rho_f\|\boldsymbol{y}\|$ under any rotation with a maximum rotation angle of $\varphi_{\max}$, i.e.

$$\boldsymbol{e}^{[-\boldsymbol{r}(t)]}\boldsymbol{y} \in B\left(\delta_f\boldsymbol{y}, \rho_f\|\boldsymbol{y}\|\right), \quad \forall t \in [0, T], \tag{42}$$

where

$$\begin{aligned} \delta_f &= \cos(\tilde{\varphi}), \\ \rho_f &= \sin(\tilde{\varphi}), \\ \tilde{\varphi} &= \min\left[\varphi_{\max}, \frac{\pi}{2}\right]. \end{aligned} \tag{43}$$

Therefore, using (42), a sufficient condition for thrust feasibility is

$$\begin{aligned} \boldsymbol{A}_f\left(\delta_f\boldsymbol{h} + \rho_f\|\boldsymbol{h}\|\boldsymbol{\Delta}\right) \preceq \boldsymbol{b_f}, \quad & \forall \boldsymbol{h} \in \mathcal{H}, \\ & \forall \boldsymbol{\Delta} \in B(0, 1), \end{aligned} \tag{44}$$

or equivalently (as all row vectors of $\boldsymbol{A}_f$ are unit vectors)

$$\boldsymbol{A}_f\delta_f\boldsymbol{h} \preceq \boldsymbol{b}_f - \rho_f\|\boldsymbol{h}\|\boldsymbol{1}, \quad \forall \boldsymbol{h} \in \mathcal{H}, \tag{45}$$

where $\boldsymbol{1} = (1, \ldots, 1)$.

Since the bounding box $\mathcal{H}$ is a convex set, every point in $\mathcal{H}$ can be written as a convex combination of the vertices of $\mathcal{H}$. Furthermore, it is straightforward to verify that also (45) describes a convex set and hence, for any two points $\{\boldsymbol{h}_1, \boldsymbol{h}_2\} \in \mathcal{H}$ that satisfy (45), any convex combination

$$\boldsymbol{h} = \lambda\boldsymbol{h}_1 + (1 - \lambda)\boldsymbol{h}_2, \quad \lambda \in [0, 1] \tag{46}$$

also satisfies (45). It is therefore sufficient to only validate that the eight vertices of the bounding box $\mathcal{H}$ satisfy (45) in order to guarantee feasibility with respect to the thrust input.

A visual interpretation of the thrust input feasibility check is illustrated in Fig. 2.

C. Angular Velocity Input Feasibility

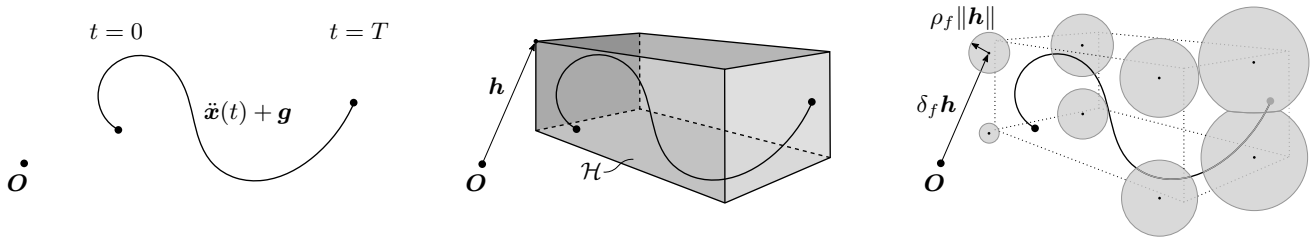
The angular velocity during the execution of the rotational motion primitive is given in (27) and has to satisfy the constraint

$$\boldsymbol{A}_\omega\boldsymbol{\omega}(t) \preceq \boldsymbol{b}_\omega, \quad \forall t \in [0, T]. \tag{47}$$

First, it is determined whether the initial angular velocity is feasible, i.e.

$$\boldsymbol{A}_\omega\boldsymbol{d}_3 \preceq \boldsymbol{b}_\omega, \tag{48}$$

and then it is verified that the angular velocity remains feasible during the entire motion primitive by computing a bounding box on the required rotation vector velocity and ensuring



(a) (b) (c)

Fig. 2: Visual interpretation of the thrust input feasibility check. Fig. 2(a) depicts the mass-normalized thrust trajectory $\ddot{\boldsymbol{x}}(t) + \boldsymbol{g}$ during the time interval $t \in [0, T]$. If the initial thrust at $t = 0$ is found to be feasible, then a bounding box $\mathcal{H}$ on the required thrust is computed (see Fig. 2(b)). Because the thrust constraint (13) is expressed in the vehicle's body frame but the bounding box $\mathcal{H}$ is computed with respect to the vehicle's initial body frame, i.e. the attitude at $t = 0$, it needs to be verified that every point $\boldsymbol{h} \in \mathcal{H}$ satisfies the thrust constraint when rotated into the vehicle's body frame for any $t \in [0, T]$. If $\varphi_{\max}$ denotes the maximum rotation angle of the rotational motion primitive during the interval $[0, T]$ relative to the vehicle's initial attitude, then it can be shown that any point $\boldsymbol{y} \in \mathbb{R}^3$ remains in a closed ball with center $\delta_f \boldsymbol{y}$ and minimal radius $\rho_f \|\boldsymbol{y}\|$ under any rotation with a maximum rotation angle of $\varphi_{\max}$. Since the bounding box $\mathcal{H}$ is convex, it is sufficient to verify that the bounding balls of the eight vertices of $\mathcal{H}$ lie in the set of attainable thrusts (see Fig. 2(c)).

that all points in the bounding box satisfy the constraints when mapped to angular velocities under any rotation with a maximum rotation angle of $\varphi_{\max}$.

Let $\mathcal{V}$ describe the bounding box of the rotation vector velocity,

$$\mathcal{V} = \{\boldsymbol{y} \in \mathbb{R}^3 \mid \boldsymbol{v}_{\min} \preceq \boldsymbol{y} \preceq \boldsymbol{v}_{\max}\}, \tag{49}$$

where the lower and upper bound $\boldsymbol{v}_{\min}$ and $\boldsymbol{v}_{\max}$ are computed by finding the minimum and maximum of the rotation vector velocity $\dot{\boldsymbol{r}}(t)$ along each axis on the interval $[0, T]$. This involves evaluating $\dot{\boldsymbol{r}}(t)$ at the boundaries of the interval and at the root of its derivative (a linear polynomial).

The constraint (47) can then be rewritten as

$$\boldsymbol{A}_\omega \boldsymbol{W}(\boldsymbol{r}(t))\, \boldsymbol{v} \preceq \boldsymbol{b}_\omega, \quad \forall \boldsymbol{v} \in \mathcal{V}, \quad \forall t \in [0, T]. \tag{50}$$

Similar to (42), it is shown in the Appendix C that any point $\boldsymbol{y} \in \mathbb{R}^3$ remains in a ball with center $\delta_\omega \boldsymbol{y}$ and minimal radius $\rho_\omega \|\boldsymbol{y}\|$ under the map $\boldsymbol{W}(\boldsymbol{r}(t))\, \boldsymbol{y}$ for any rotation with a maximum rotation angle of $\varphi_{\max}$, i.e.

$$\boldsymbol{W}(\boldsymbol{r}(t))\, \boldsymbol{y} \in B(\delta_\omega \boldsymbol{y}, \rho_\omega \|\boldsymbol{y}\|), \quad \forall t \in [0, T], \tag{51}$$

where

$$\begin{aligned} \delta_\omega &= \begin{cases} 1, & \text{if } \varphi_{\max} = 0, \\ \frac{\sin(\tilde{\varphi})}{\tilde{\varphi}}, & \text{otherwise}, \end{cases} \\ \rho_\omega &= \begin{cases} 0, & \text{if } \varphi_{\max} = 0, \\ \frac{1-\cos(\tilde{\varphi})}{\tilde{\varphi}}, & \text{otherwise}, \end{cases} \\ \tilde{\varphi} &= \min[\varphi_{\max}, \overline{\varphi}], \end{aligned} \tag{52}$$

where $\overline{\varphi}$ is defined as the rotation angle at which the ball's radius is maximized,

$$\overline{\varphi} := \arg\max_{\varphi > 0} \frac{1 - \cos(\varphi)}{\varphi}. \tag{53}$$

It therefore follows from (51) that a sufficient condition to establish feasibility is

$$\begin{aligned} \boldsymbol{A}_\omega \left(\delta_\omega \boldsymbol{v} + \rho_\omega \|\boldsymbol{v}\| \boldsymbol{\Delta}\right) \preceq \boldsymbol{b}_\omega, \quad & \forall \boldsymbol{v} \in \mathcal{V}, \\ & \forall \boldsymbol{\Delta} \in B(0, 1), \end{aligned} \tag{54}$$

or equivalently

$$\boldsymbol{A}_\omega \delta_\omega \boldsymbol{v} \preceq \boldsymbol{b}_\omega - \rho_\omega \|\boldsymbol{v}\| \boldsymbol{1}, \quad \forall \boldsymbol{v} \in \mathcal{V}. \tag{55}$$

As for the verification of the thrust feasibility, it is sufficient to only verify that all vertices of the bounding box $\mathcal{V}$ satisfy (55) in order to determine feasibility with respect to the angular velocity input since both $\mathcal{V}$ and (55) are convex sets.

D. Discussion

The motion primitives' feasibility can be tested at little computational cost as it only involves verifying the feasibility of a few distinct points, namely the vertices of the bounding boxes $\mathcal{H}$ and $\mathcal{V}$. Computing these vertices as well as computing the maximum rotation angle only requires finding the roots of a polynomial of order at most four and can therefore be done in closed-form.

Furthermore, using the same methods as shown above, the feasibility checks can easily be extended to handle state constraints or further input constraints of the form

$$\boldsymbol{A}_s \boldsymbol{s}(t) \preceq \boldsymbol{b}_s \tag{56}$$

or

$$\boldsymbol{A}_s \boldsymbol{R}(t) \boldsymbol{s}(t) \preceq \boldsymbol{b}_s, \tag{57}$$

where $\boldsymbol{s}(t)$ can be any state or control input trajectory (or linear combinations thereof), as computing a bounding box on these only involves finding the roots of at most a quartic polynomial. The constraints (56) and (57) can be used, for example, to encode boundaries on the vehicle's position or maximum angular velocity expressed in the inertial frame.

Due to the conservative approximations taken in the input feasibility tests (45) and (55) by computing, for example, bounding boxes, the input feasibility tests are sufficient but not necessary. In particular, applying the input feasibility tests can result in three possible outcomes. The motion primitives can either be

- provably infeasible, if either (38) or (48) is false,
- provably feasible, if (45) and (55) hold true for all vertices of the corresponding bounding box,
- or otherwise, their feasibility is indeterminable.

VII. Trajectory Generation Algorithm

This section describes the trajectory generation algorithm consisting of the previously derived motion primitives and feasibility checks. First, a translational and rotational motion primitive on the interval $\mathcal{T} = [\tau_1, \tau_2] = [0, T]$ are planned from a given initial state to a given final state. Afterwards, the feasibility of the motion primitives, or more precisely of the corresponding control input trajectories $\boldsymbol{f}(t)$ and $\boldsymbol{\omega}(t)$, is tested. If the feasibility tests return that the motion primitives are either feasible or infeasible, the algorithm terminates accordingly. If the feasibility could not be determined, the time interval is split in half,

$$\tau_{\frac{1}{2}} = \frac{\tau_1 + \tau_2}{2}, \tag{58}$$

$$\mathcal{T}_1 = [\tau_1, \tau_{\frac{1}{2}}], \quad \mathcal{T}_2 = [\tau_{\frac{1}{2}}, \tau_2]. \tag{59}$$

If the new interval length $\tau_{\frac{1}{2}} - \tau_1$ is below some threshold τ_{min}, the algorithm terminates without being able to generate a feasible trajectory. Otherwise, the algorithm is applied recursively on the subinterval $\mathcal{T}_1$ to generate a trajectory from the initial state $\boldsymbol{s}(\tau_1)$ to the final state $\boldsymbol{s}(\tau_{\frac{1}{2}})$, where $\boldsymbol{s}(t) = (\boldsymbol{x}(t), \dot{\boldsymbol{x}}(t), \boldsymbol{R}(t))$ denotes the state trajectory corresponding to the motion primitives computed on the interval $\mathcal{T}$. If the outcome is feasible, then the algorithm is also applied recursively on the second subinterval $\mathcal{T}_2$ to generate a trajectory from the initial state $\boldsymbol{s}(\tau_{\frac{1}{2}})$ to the final state $\boldsymbol{s}(\tau_2)$. If the outcome on the interval $\mathcal{T}_2$ is also feasible, the motion primitives of the two intervals are concatenated and the algorithm terminates with a proven feasible trajectory. In addition to the initial state of the trajectory on $\mathcal{T}_2$ being equal to the final state of the trajectory on $\mathcal{T}_1$, also the initial and final acceleration and angular velocities on $\mathcal{T}_2$ and $\mathcal{T}_1$ are set to be equal, in particular to $\ddot{\boldsymbol{x}}(\tau_{\frac{1}{2}})$ and $\boldsymbol{\omega}(\tau_{\frac{1}{2}})$, in order to ensure smooth transitions on the control inputs when concatenating the motion primitives. The initial and final acceleration and angular velocity on the interval $\mathcal{T}$ are chosen likewise and are, for example, set to zero for rest-to-rest maneuvers.

Bisecting the time interval on which the trajectories are generated is motivated by the conservative approximation of the feasibility constraints. If the trajectory is planned on a shorter time interval, the corresponding maximum rotation angle will be smaller and the bounding boxes will be tighter, hence reducing the conservativeness of the feasibility checks.

Note that due to the principle of optimality (see for example [26]), the translational motion primitive only needs to be computed once, as the optimal position trajectory $\boldsymbol{x}(t),\ t \in \mathcal{T}$, is also optimal on the subintervals $t \in \mathcal{T}_1$ and $t \in \mathcal{T}_2$, respectively. When verifying the motion primitives' feasibility, the fact that the rotational motion primitive is planned from $\boldsymbol{r}(0) = 0$ is exploited in order to efficiently compute the maximum rotation angle. However, this requires the nonlinear transformations of the boundary conditions (25) and (32) and as a consequence, the principle of optimality cannot be applied to the rotational motion primitives. The optimal attitude trajectory computed on a subinterval $[\tau_1, \tau_2] \subset \mathcal{T}$ generally differs from the corresponding attitude trajectory computed on the entire interval $\mathcal{T}$, in particular if $\tau_1 \neq 0$, and therefore needs to be recomputed (see Fig. 3). As a result of the changing attitude trajectories due to bisection, a trajectory that is feasible but whose feasibility cannot be determined may be rendered infeasible in a subsequent recursion step.

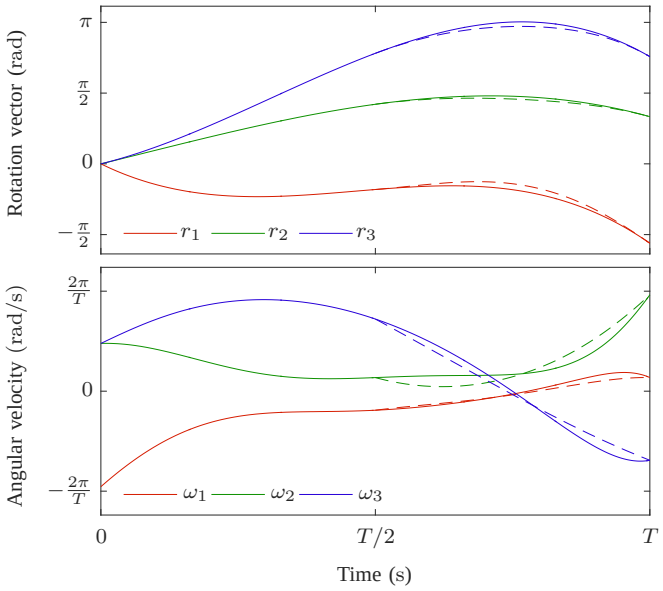


Fig. 3: Attitude and angular velocity trajectories obtained by applying the proposed trajectory generation algorithm recursively on the interval $[0, T]$. The solid line represents the trajectories computed on the intervals $[0, T]$ and $[0, T/2]$, respectively. The dashed line represents the trajectories computed on the interval $[T/2, T]$. For ease of interpretation, the rotation vector trajectory on the interval$[T/2, T]$ is rotated by the attitude at $t = T/2$. It can be seen that the trajectories computed on the interval $[T/2, T]$ differ from the corresponding trajectories computed on the entire time interval, despite having the same initial and final attitude and angular velocity.

VIII. Performance Evaluation

This section attempts to assess the performance of the proposed trajectory generation algorithm with respect to the design objective of rapidly computing a large set of feasible state-to-state trajectories for a given maneuver duration in real time. If the trajectory generation algorithm finds a trajectory, it is guaranteed to satisfy the system dynamics and input constraints. However, if the algorithm is unable to find a feasible trajectory, it does not imply that no feasible trajectory exists, but could be due to

- the restriction of the trajectories to the motion primitives of Section V (and concatenations thereof),
- or to the conservative approximations of the input constraints when verifying the feasibility, as the derived feasibility conditions are only sufficient but not necessary and therefore may require to perform a bisection step that will render the trajectories infeasible.

In order to evaluate the conservativeness of the algorithm and its applicability to real-time applications, its computational costs when implemented on a standard laptop computer are presented and a comparison with time-optimal trajectories is given. The evaluations are carried out for a minimum time interval length of $\tau_{\text{min}} = 0.01\text{s}$ and for the omni-directional octorotor vehicle presented in [6], whose thrust is constrained

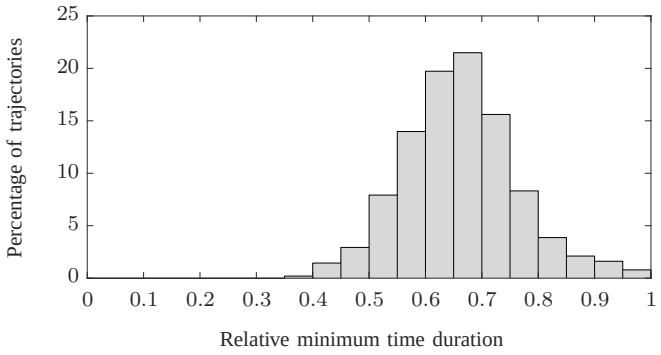


Fig. 4: Comparison of the time-optimal maneuver duration relative to the shortest maneuver duration for which the presented algorithm is able to find a feasible trajectory. The comparison is carried out for 10 000 trajectories starting from rest and steering the vehicle to randomized end states.

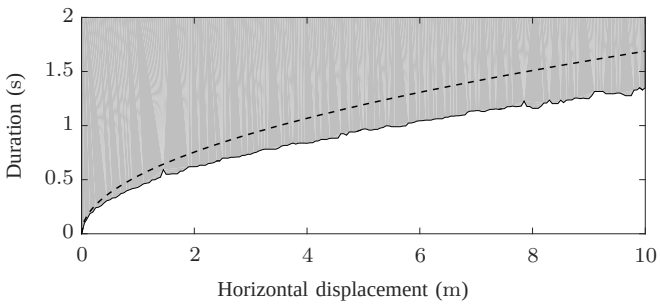


Fig. 5: The grey area represents the set of horizontal displacements along the x_1-axis and end times for which feasible trajectories exist. This set was found by computing the time-optimal maneuver numerically (solid line) and hence a trajectory longer than this always exists by simply waiting after executing the time-optimal maneuver, but by definition no shorter trajectory exists (white area). The dashed line indicates the shortest maneuver durations for which the algorithm was able to generate a feasible trajectory.

to a rhombic dodecahedron with an inradius of $19.6\mathrm{m/s^2}$ and whose angular velocity components are constrained to $|\omega_i| \leq 3\mathrm{rad/s},\ i \in \{1,2,3\}$ (see Appendix A for more details).

A. Computational Performance

In the following, the average computation time required to run the algorithm on a standard laptop computer with an Intel Core i7-3840QM CPU running at $2.8\,\mathrm{GHz}$, with $16\,\mathrm{GB}$ of RAM, is given. The algorithm was compiled with Microsoft Visual C++ 14.0 and was run as a single thread application. The average computation time was evaluated through generating one billion trajectories starting from rest at the origin to randomized final states with maneuver durations drawn uniformly at random between $0.25\,\mathrm{s}$ and $10.0\,\mathrm{s}$. The vehicle's initial attitude was chosen such that its body frame is aligned with the inertial frame. The final positions and velocities were chosen uniformly at random from a box centered at the origin with side lengths of $10\,\mathrm{m}$ and $10\,\mathrm{m/s}$, respectively, and the final attitudes where chosen uniformly at random from $\mathrm{SO}(3)$. Since the maneuvers were planned from rest, the initial acceleration and angular velocity were set to zero. The components of the final acceleration and angular velocity were chosen uniformly at random between $-5\,\mathrm{m/s^2}$ and $5\,\mathrm{m/s^2}$, and $-1.5\,\mathrm{rad/s}$ and $1.5\,\mathrm{rad/s}$, respectively.

The average computation time per trajectory, including verifying its input feasibility, was measured to be $1.94\,\mu\mathrm{s}$. Of all computed trajectories, 86.3% were proven to be feasible, 10.9% were found to be infeasible, and the feasibility of the remaining 2.8% could not be determined. Although the algorithm was implemented on a laptop computer, modern flight computers such as the Qualcomm Snapdragon Flight[2] have similar specifications and the algorithm could therefore also be run on board at a comparable rate.

The low computational cost of the proposed trajectory generation algorithm can be exploited to rapidly search over a large set of end states and maneuver durations that achieve a certain high level goal, and to select the trajectory that minimizes some high level cost.

[2]https://developer.qualcomm.com/hardware/snapdragon-flight

B. Comparison with Time-Optimal Trajectories

Consider the motion primitives of Section V that guide the vehicle from an initial state to a desired end state in time T. As the maneuver duration is increased, $T \to \infty$, it can be derived from (19) and (30) that the motion primitives' coefficients $\boldsymbol{c}_1$, $\boldsymbol{c}_2$, $\boldsymbol{c}_3$, $\boldsymbol{d}_1$ and $\boldsymbol{d}_2$ converge to zero. The vehicle's required control inputs $\boldsymbol{f}(t)$ and $\boldsymbol{\omega}(t)$ therefore converge to

$$\lim_{T\to\infty} \boldsymbol{f}(t) = \boldsymbol{e}^{[-\boldsymbol{\omega}_0 t]} \boldsymbol{R}_0^T \left(\boldsymbol{a_0} + \boldsymbol{g}\right), \tag{60}$$

$$\lim_{T\to\infty} \boldsymbol{\omega}(t) = \boldsymbol{\omega}_0. \tag{61}$$

The initial acceleration $\boldsymbol{a}_0$ and angular velocity $\boldsymbol{\omega}_0$ are design parameters that can be chosen arbitrarily. The algorithm is therefore guaranteed to find a feasible trajectory for sufficiently long maneuver durations if the multirotor vehicle can generate sufficient thrust in any direction to overcome gravity $\boldsymbol{g}$, as for example the multirotor vehicles [5], [6]. In order to quantify the algorithm's conservatism, the minimum maneuver duration for which the algorithm can find a feasible trajectory is compared with the duration of the time-optimal maneuver.

The time-optimal trajectories are computed using GPOPS-II [29], a numerical optimal control software, and the fastest trajectory found by the proposed algorithm is used as initial guess for the time-optimal maneuver. Fig. 4 shows the result of a comparison for 10 000 maneuvers starting from rest at the origin and with the vehicle's body frame aligned with the inertial frame to final states drawn uniformly at random from the set described in Section VIII-A. On average, the time-optimal maneuver duration is 33.8% faster than the shortest maneuver for which the presented algorithm is able to find a feasible trajectory[3].

A comparison of the maneuver durations for pure rest-to-rest horizontal translations along the x_1-axis with the initial and final body frame aligned with the inertial frame is given in

[3]A bisection method was used to determine the shortest maneuver duration for which the presented trajectory generation algorithm is able to find a feasible trajectory. Computing the shortest maneuver duration using bisection took on average less than one millisecond (bisection was terminated when the bisection interval size was smaller than one millisecond), while GPOPS-II took between several minutes and an hour to compute a solution.

Fig. 5. The time-optimal maneuvers are on the order of 20% faster than the fastest feasible trajectories generated by the proposed algorithm. This is due to limiting the trajectories to the motion primitives of Section V, but also to the decoupled planning of the motion primitives. An analysis of the time-optimal maneuvers revealed that it is beneficial for the vehicle to rotate in order to accelerate faster and hence reach the target position quicker, even though the initial and final attitude are identical.

IX. Experimental Results

This section presents an experiment to verify the feasibility of the trajectory generation algorithm for real-time applications that require the evaluation of a large set of possible trajectories. The goal is for a fully-actuated multirotor vehicle to catch a ball thrown by a person in a small pouch. This task was chosen because a large number of trajectories exist that achieve the goal and need to be evaluated in real time, and is therefore well suited to test the algorithm's speed and assess its performance experimentally.

The experiments were carried out in the Flying Machine Arena [20], an indoor aerial vehicle test bed at ETH Zurich, using the omni-directional octorotor vehicle [6]. An overhead motion-capture system provides position and attitude information of the vehicle and the ball. This information is processed on a desktop computer that estimates the state of the vehicle and the ball, plans a catching maneuver accordingly and sends out control commands to the vehicle through a low-latency radio link at a rate of $50\,\mathrm{Hz}$. In order to be able to catch the ball, the octorotor vehicle is equipped with a pouch of radius $5.75\,\mathrm{cm}$ at a distance of $47.5\,\mathrm{cm}$ from the vehicle's center of mass (see Fig. 6). A table tennis ball, modelled as a point mass with aerodynamic drag proportional to its speed squared, is used in the experiments. All results shown in the following are from actual flight experiments.

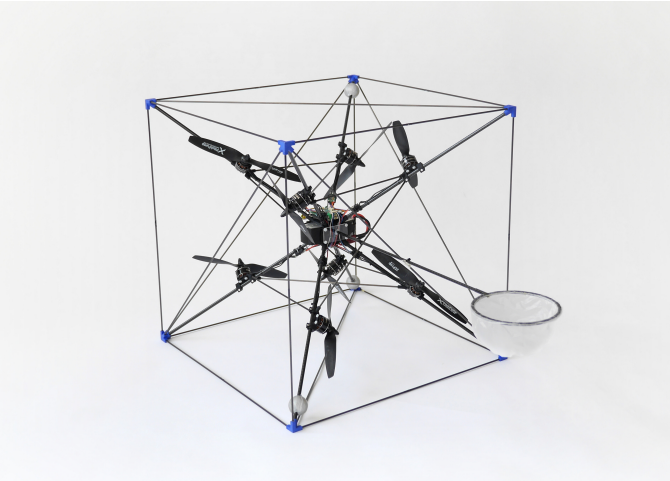


Fig. 6: Omni-directional octorotor vehicle equipped with a pouch. The pouch is used to catch a thrown ball and thereby demonstrating the algorithm's performance experimentally.

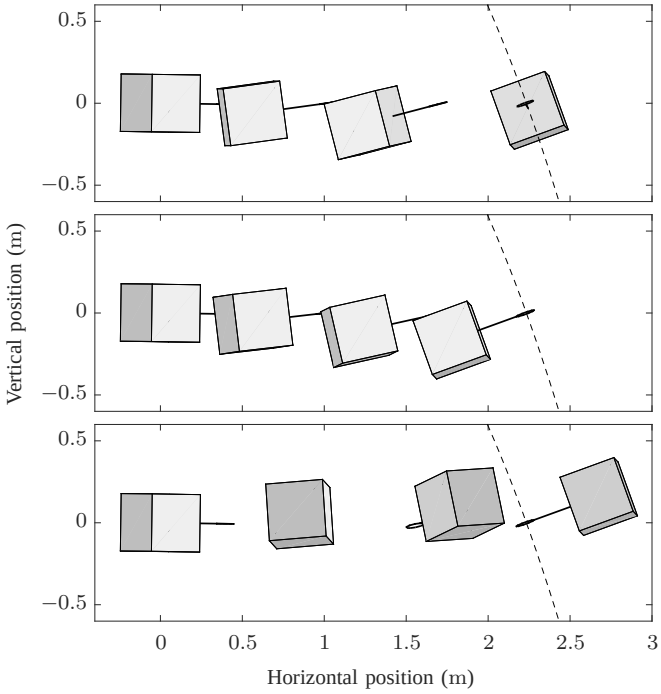


Fig. 7: Three candidate catching maneuvers that guide the octorotor vehicle from rest to a state in which the ball is caught in the pouch. The dashed line represents the predicted ball flight path, and the goal is to catch the ball at a height of zero meters. The maneuver shown in the top plot was found to be the feasible maneuver with the lowest cost, and the maneuver on the bottom was found to be infeasible.

A. High Level Path Planner

The trajectory generation algorithm is embedded in a high level path planner that performs a brute-force search over a set of possible catching and stopping maneuvers and selects the maneuvers with the lowest cost.

1) Catching Maneuver: As soon as the ball is thrown into the air, its catching position and flight duration is predicted and a catching maneuver is planned using the current vehicle state as initial condition and with free initial acceleration and angular velocity (see Appendix B). The catching position is defined to be the position where the ball crosses a fixed height. The vehicle's state at the end of the catching maneuver is chosen such that the pouch is at the catching position and that its normal direction points in the opposite direction of the ball's velocity. In addition, the ball is caught with the pouch at rest except for an angular velocity about its normal direction in order to be less prone to timing errors. A brute-force search over 1600 end states is performed, in particular over

- 40 final positions, equally spaced around the ball's catching position,
- and 40 final angular velocities, equally spaced between $\pm 3\,\mathrm{rad/s}$ and with the vehicle's velocity and acceleration set accordingly such that the pouch remains at rest.

Fig. 7 shows three of the possible 1600 catching maneuvers that are evaluated for an initial prediction of the ball's flight path.

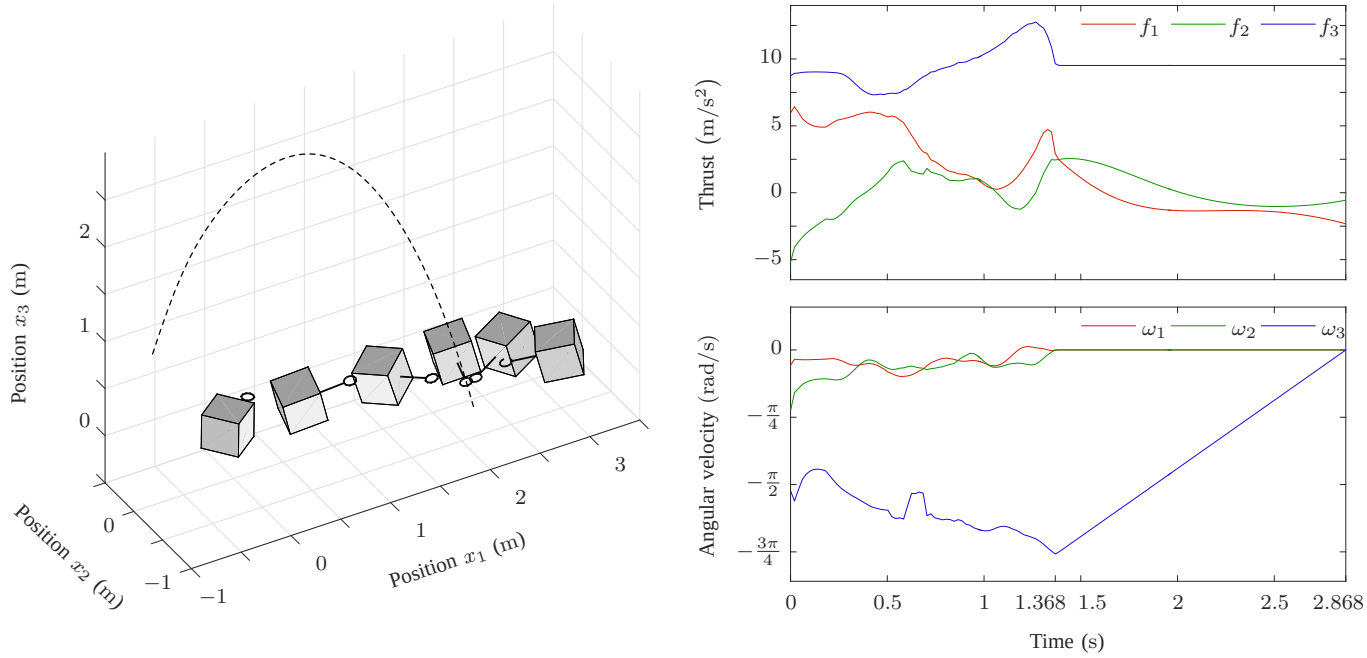


Fig. 8: The left hand side shows a complete catching and stopping maneuver. The dashed line denotes the ball's flight path. After the ball is first detected, the vehicle has $1.368\,\mathrm{s}$ time to move the pouch to the catching position starting from rest. Snapshots of the catching maneuver are depicted at the time instances $t = \{0\,\mathrm{s}, 0.62\,\mathrm{s}, 0.96\,\mathrm{s}, 1.368\,\mathrm{s}, 1.82\,\mathrm{s}, 2.868\,\mathrm{s}\}$. During the catching maneuver, the octorotor vehicle translated $2.25\,\mathrm{m}$ and rotated $72.2°$, and had a final angular velocity of $-2.38\,\mathrm{rad/s}$ at the time of catching. The right hand side shows the respective control inputs. Note that the control inputs during the catching maneuver do not represent a single catching trajectory but are the result of constantly re-planning the catching maneuver at each controller update step.

The feasibility of the trajectories is verified with respect to the vehicle's input constraints as introduced in Section VIII and presented in more detail in Appendix A. Furthermore, a box constraint on the position using (56) was implemented to ensure that the vehicle remains within a safe flying space.

From all feasible catching maneuvers for which also a stopping maneuver exists, the catching maneuver with the lowest cost

$$\tilde{J}_{\text{trans}} + \alpha \tilde{J}_{\text{rot}} \tag{62}$$

is selected, where $\tilde{J}_{\text{trans}}$ and $\tilde{J}_{\text{rot}}$ are the sums of the translational and rotational costs (22) and (33) over all concatenated motion primitives. The design parameter α describes a trade-off between translational and rotational costs and is set to $\alpha = 10$ in the experiments.

2) Stopping Maneuver: Using the final state of the catching maneuver as initial condition, a stopping maneuver is planned that guides the vehicle to rest with free final position and attitude. The adapted translational and rotational motion primitive with the final position and attitude being subject to optimization are given in Appendix B. A brute-force search over five trajectories with a maneuver duration equally spaced between $0.5\,\mathrm{s}$ and $1.5\,\mathrm{s}$ is performed and the feasible stopping maneuver with the lowest cost (62) is selected.

B. Control Strategy

During the catching maneuver, the output of the high level path planner is applied as an implicit feedback law. At each controller update step, i.e. every $20\,\mathrm{ms}$, the prediction of the catching position and the remaining flight duration is refined using the most recent motion capture data. The catching maneuver is then re-planned starting from the vehicle's current state by evaluating a maximum of 8000 ($40 \times 40 \times 5$) trajectories and the initial control inputs $\boldsymbol{f}$ and $\boldsymbol{\omega}$ corresponding to the feasible catching maneuver with the lowest cost are sent to the vehicle. If no feasible trajectory can be found, then the last found feasible trajectory is tracked using the feedback controller presented in [6]. The same feedback controller is also used to track the stopping maneuver.

C. Results

The full catching and stopping maneuver for the same throw as shown in Fig. 7 is depicted in Fig. 8. The ball's flight lasted $1.368\,\mathrm{s}$, during which the vehicle started from rest and covered a distance of $2.25\,\mathrm{m}$ and rotated $72.2°$ in order to catch the ball. The catching maneuver is similar to the maneuver shown in the top plot of Fig. 7 for the initial prediction of the ball's flight path, but is continuously re-planned as new pose information of the vehicle and the ball become available. During the catching maneuver, a total of 120 785 trajectories were computed, 110 400 catching maneuvers and 10 385 stopping maneuvers.

A video showing the vehicle catching the ball is attached to the paper and can also be accessed on `https://youtu.be/0gR1ekapOAE`.

X. Conclusion

This paper presented and analyzed a computationally efficient method for the rapid generation and feasibility verification of trajectories that guide a fully-actuated multirotor vehicle from any initial to any desired end state in a given time. The algorithm is based on recursively generating state-to-state motion primitives and subsequently verifying their feasibility.

Computationally inexpensive motion primitives for the vehicle's position and attitude were derived that are characterized by polynomials in time. This allowed for efficiently computing bounds on the vehicle's acceleration, rotation angle and rotation vector velocity, which were then used to verify the motion primitives' feasibility. A trajectory generation algorithm based on the derived motion primitives and feasibility tests was introduced. If the feasibility of a motion primitive cannot be determined due to the conservative approximation of the constraints, the trajectory generation algorithm is applied recursively on subintervals and the resulting motion primitives are concatenated.

The algorithm's computational efficiency and speed is due to the restriction of the trajectories to certain motion primitives for which closed-form solutions exist and due to the conservative approximation of the input constraints when testing the feasibility. However, this comes at the expense of not always being able to find a feasible trajectory for a given initial and final state and maneuver duration although a feasible trajectory exists. In order to characterize its performance and assess its conservativeness, the trajectories generated by the algorithm were compared with time-optimal maneuvers.

The feasibility of the algorithm for real-time applications that require the evaluation of large numbers of trajectories was verified experimentally by catching a ball thrown by a person with a small pouch attached to an omni-directional octorotor vehicle. For this purpose, the algorithm was embedded in a high level path planner that performed a brute-force search over a large number of candidate trajectories and selected the maneuver with the lowest cost. In order to cope with disturbances and changing final states, the trajectory planning was repeated at each controller update step with the vehicle's current state as initial condition, and the output trajectory was then applied as an implicit feedback law.

While the presented method allows for generating state-to-state trajectories for any fully-actuated multirotor vehicle, it is only guaranteed to find a feasible trajectory for an infinitely large maneuver duration if the vehicle is able to hover at any attitude. This is due to the decoupled planning of the translational and rotational motion primitives which does not take the input constraints into account. Considering the algorithm's speed, one approach to overcome this limitation could be to embed the trajectory generation algorithm in a high level path planner such as a rapidly exploring random tree algorithm in order to to explore different attitude paths and find feasible trajectories.

Appendix A
Octorotor Vehicle Input Constraints

This appendix presents the input constraints of the omni-directional octorotor vehicle presented in [6] that is used in this paper as an example fully-actuated multirotor vehicle to evaluate the performance of the trajectory generation algorithm as well as for the experimental results.

The octorotor vehicle uses reversible fixed-pitch rotors that can produce both positive and negative thrust. Each single rotor thrust is subject to the saturation limits

$$-f_{\text{max}} \leq f_{\text{rot}} \leq f_{\text{max}}. \tag{63}$$

If the aerodynamic interference between rotors is neglected, the collective thrust $\boldsymbol{f}$ and torque $\boldsymbol{t}$ produced by the eight rotors can be written as

$$\begin{bmatrix} \boldsymbol{f} \\ \boldsymbol{t} \end{bmatrix} = \underbrace{\begin{bmatrix} \boldsymbol{n}_1 & \dots & \boldsymbol{n}_8 \\ \boldsymbol{p}_1 \times \boldsymbol{n}_1 + \kappa \boldsymbol{n}_1 & \dots & \boldsymbol{p}_8 \times \boldsymbol{n}_8 + \kappa \boldsymbol{n}_8 \end{bmatrix}}_{=:\boldsymbol{B}} \begin{bmatrix} f_{\text{rot},1} \\ \vdots \\ f_{\text{rot},8} \end{bmatrix}, \tag{64}$$

where $f_{\text{rot},i}$ is the thrust magnitude generated by the i-th rotor, $\boldsymbol{n}_i$ is the rotor disk normal, $\boldsymbol{p}_i$ is the rotor position relative to the vehicle's center of mass and κ is the rotor's specific thrust-to-drag ratio. The exact numerical values of the rotor configuration can be found in [6]. The set of attainable thrusts and torques can be written as

$$\{\boldsymbol{B}\boldsymbol{f}_{\text{rot}} \in \mathbb{R}^6 \mid \boldsymbol{f}_{\text{rot}} \in \mathbb{R}^8, \|\boldsymbol{f}_{\text{rot}}\|_\infty \leq f_{\text{max}}\}. \tag{65}$$

Applying the method presented in [30], the set of attainable thrusts and torques (65) can be described by a polyhedron of the form

$$\begin{bmatrix} \tilde{\boldsymbol{A}}_f & \tilde{\boldsymbol{A}}_\tau \end{bmatrix} \begin{bmatrix} \boldsymbol{f} \\ \boldsymbol{t} \end{bmatrix} \preceq \tilde{\boldsymbol{b}}. \tag{66}$$

A slice through this polyhedron at $\boldsymbol{t} = 0$ then yields the set of attainable thrusts when producing zero torque

$$\tilde{\boldsymbol{A}}_f \boldsymbol{f} \preceq \tilde{\boldsymbol{b}}, \tag{67}$$

and can be further simplified by removing all redundant inequalities, resulting in the rhombic dodecahedron

$$\boldsymbol{A}_f \boldsymbol{f} \preceq \boldsymbol{b}_f, \tag{68}$$

with

$$\boldsymbol{A}_f = \frac{1}{2} \begin{bmatrix} -2 & 0 & 0 \\ -1 & 1 & \sqrt{2} \\ -1 & 1 & -\sqrt{2} \\ -1 & -1 & \sqrt{2} \\ -1 & -1 & -\sqrt{2} \\ 0 & -2 & 0 \\ 0 & 2 & 0 \\ 1 & 1 & \sqrt{2} \\ 1 & 1 & -\sqrt{2} \\ 1 & -1 & \sqrt{2} \\ 1 & -1 & -\sqrt{2} \\ 2 & 0 & 0 \end{bmatrix} \begin{bmatrix} \cos\left(\frac{\pi}{12}\right) & -\sin\left(\frac{\pi}{12}\right) & 0 \\ \sin\left(\frac{\pi}{12}\right) & \cos\left(\frac{\pi}{12}\right) & 0 \\ 0 & 0 & 1 \end{bmatrix}, \tag{69}$$

$$\boldsymbol{b}_f = f_{\text{max}} \sqrt{\frac{32}{3}} \mathbf{1}. \tag{70}$$

The set of attainable thrusts (68) for a mass-normalized maximum rotor thrust of $f_{\text{max}} = 6\,\text{m/s}^2$ is illustrated in Fig. 9.

Rotating the octorotor vehicle while maintaining a thrust in the same inertial direction typically requires some rotors to change their spinning direction. However, reversing the spinning direction of too many motors in rapid succession

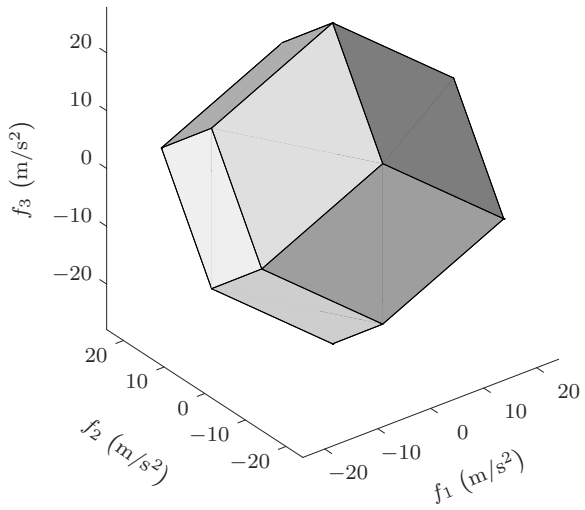


Fig. 9: Rhombic dodecahedron with an inradius of $19.6\,\mathrm{m/s^2}$ describing the set of attainable thrusts for the octorotor vehicle presented in [6].

can cause instabilities. Experiments revealed that the low-order dynamic model with angular velocity as the control input approximates well the true system dynamics for an angular velocity of up to $\omega_{\text{max}} = 3\,\mathrm{rad/s}$ per axis. The angular velocity of the vehicle is thus constraint to the box

$$\boldsymbol{A}_\omega \boldsymbol{\omega} \preceq \boldsymbol{b}_\omega, \tag{71}$$

with

$$\boldsymbol{A}_\omega = \begin{bmatrix} 1 & 0 & 0 \\ -1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & -1 \end{bmatrix}, \tag{72}$$

$$\boldsymbol{b}_\omega = \omega_{\text{max}} \mathbf{1}. \tag{73}$$

Appendix B
Derivation of Motion Primitives

This appendix presents the derivations of the translational and rotational motion primitive introduced in Section V. The motion primitives' trajectories are computed using Pontryagin's minimum principle and solutions are provided for the different initial and final conditions used in this paper.

A. Translational Motion Primitive

We seek to find the motion primitive that minimizes the cost

$$J_{\text{trans}} = \frac{1}{T} \int_0^T \|\dddot{\boldsymbol{x}}(t)\|^2 dt. \tag{74}$$

The optimal jerk $\dddot{\boldsymbol{x}}(t)$ minimizing (74) can be computed for each axis individually as the total cost is simply the sum of the costs along each axis,

$$J_{\text{trans}} = \sum_{i=1}^{3} J_{\text{trans},i} = \sum_{i=1}^{3} \frac{1}{T} \int_0^T \dddot{x}_i^2(t) dt. \tag{75}$$

In the following, the solution along the x_i-direction, $i \in \{1,2,3\}$, is derived. For the sake of readability, the subscript i is omitted in the following and, for example, the scalar x denotes x_i, i.e. the i-th component of $\boldsymbol{x}$. Let $\boldsymbol{s} = (s_1, s_2, s_3) = (x, \dot{x}, \ddot{x})$ be the state and $u = \dddot{x}$ be the input. The Hamiltonian is then defined by

$$H(\boldsymbol{s}, u, \boldsymbol{\lambda}) = \frac{1}{T} u^2 + \lambda_1 s_2 + \lambda_2 s_3 + \lambda_3 u, \tag{76}$$

where $\boldsymbol{\lambda}$ are the adjoint variables that must satisfy

$$\dot{\boldsymbol{\lambda}} = -\nabla_{\boldsymbol{s}} H(\boldsymbol{s}, u, \boldsymbol{\lambda}) = (0, -\lambda_1, -\lambda_2). \tag{77}$$

The result of the adjoint differential equation is

$$\lambda_1(t) = -\frac{1}{T} 2c_1, \tag{78}$$

$$\lambda_2(t) = \frac{1}{T}(2c_1 t + 2c_2), \tag{79}$$

$$\lambda_3(t) = \frac{1}{T}(-c_1 t^2 - 2c_2 t - 2c_3). \tag{80}$$

The optimal control u^* that minimizes the cost function is obtained by

$$u^*(t) = \arg\min_u H(\boldsymbol{s}, u, \boldsymbol{\lambda}), \tag{81}$$

$$= \frac{c_1}{2} t^2 + c_2 t + c_3, \tag{82}$$

with the corresponding optimal state trajectories

$$s_1^*(t) = \tfrac{c_1}{120} t^5 + \tfrac{c_2}{24} t^4 + \tfrac{c_3}{6} t^3 + \tfrac{c_4}{2} t^2 + c_5 t + c_6, \tag{83}$$

$$s_2^*(t) = \tfrac{c_1}{24} t^4 + \tfrac{c_2}{6} t^3 + \tfrac{c_3}{2} t^2 + c_4 t + c_5, \tag{84}$$

$$s_3^*(t) = \tfrac{c_1}{6} t^3 + \tfrac{c_2}{2} t^2 + c_3 t + c_4. \tag{85}$$

Although Pontryagin's minimum principle is only a necessary condition for optimality, it can be shown that u^* indeed minimizes (74) by exploiting the fact that the state dynamics are linear and the cost is convex [31].

If the initial and final state are fully defined, i.e. $\boldsymbol{s}(0) = (p_0, v_0, a_0)$ and $\boldsymbol{s}(T) = (p_T, v_T, a_T)$, the coefficients $c_1, \ldots, c_6$ can be determined by evaluating the optimal state trajectory $\boldsymbol{s}^*(t)$ at time $t = 0$,

$$\begin{bmatrix} c_4 \\ c_5 \\ c_6 \end{bmatrix} = \begin{bmatrix} a_0 \\ v_0 \\ p_0 \end{bmatrix}, \tag{86}$$

and at time $t = T$,

$$\begin{bmatrix} c_1 \\ c_2 \\ c_3 \end{bmatrix} = \frac{1}{T^5} \begin{bmatrix} 720 & -360T & 60T^2 \\ -360T & 168T^2 & -24T^3 \\ 60T^2 & -24T^3 & 3T^4 \end{bmatrix} \begin{bmatrix} \Delta p \\ \Delta v \\ \Delta a \end{bmatrix}, \tag{87}$$

where

$$\begin{bmatrix} \Delta p \\ \Delta v \\ \Delta a \end{bmatrix} = \begin{bmatrix} p_T - p_0 - v_0 T - \frac{1}{2} a_0 T^2 \\ v_T - v_0 - a_0 T \\ a_T - a_0 \end{bmatrix}. \tag{88}$$

By combining the optimal state trajectories (83)-(85) of each axis, the translational motion primitive's position, velocity and acceleration trajectory (16)-(18) are obtained.

If a component of the initial or final state is left free and subject to optimization, then its corresponding adjoint variable must be equal to zero at time $t = 0$ or $t = T$, respectively. The solutions for the different boundary conditions used in this paper are given in the following. In any case, the trajectories (16)-(18) and cost (22) as introduced in Section V-A remain valid.

State-To-State with Free Initial Acceleration:

$$\begin{bmatrix} c_3 \\ c_5 \\ c_6 \end{bmatrix} = \begin{bmatrix} 0 \\ v_0 \\ p_0 \end{bmatrix}, \tag{89}$$

$$\begin{bmatrix} c_1 \\ c_2 \\ c_4 \end{bmatrix} = \frac{1}{3T^5} \begin{bmatrix} 960 & -600T & 120T^2 \\ -360T & 216T^2 & -36T^3 \\ 20T^3 & -8T^4 & T^5 \end{bmatrix} \begin{bmatrix} \Delta p \\ \Delta v \\ \Delta a \end{bmatrix}, \tag{90}$$

where

$$\begin{bmatrix} \Delta p \\ \Delta v \\ \Delta a \end{bmatrix} = \begin{bmatrix} p_T - p_0 - v_0 T \\ v_T - v_0 \\ a_T \end{bmatrix}. \tag{91}$$

State-To-Rest ($v_T = a_T = 0$) with Free Final Position:

$$\begin{bmatrix} c_6 \\ c_5 \\ c_4 \end{bmatrix} = \begin{bmatrix} p_0 \\ v_0 \\ a_0 \end{bmatrix}, \tag{92}$$

$$\begin{bmatrix} c_1 \\ c_2 \\ c_3 \end{bmatrix} = \frac{1}{T^3} \begin{bmatrix} 0 & 0 \\ 12 & -6T \\ -6T & 2T^2 \end{bmatrix} \begin{bmatrix} v_0 + a_0 T \\ a_0 \end{bmatrix}. \tag{93}$$

B. Rotational Motion Primitive

We seek to find the motion primitive that minimizes the cost

$$J_{\text{rot}} = \frac{1}{T} \int_0^T \|\ddot{\boldsymbol{r}}(t)\|^2 dt. \tag{94}$$

As with the translational motion primitive, the optimal rotation vector acceleration $\ddot{\boldsymbol{r}}(t)$ minimizing (94) can be computed for each axis separately and is derived in the following for the r_i-coordinate, $i \in \{1, 2, 3\}$. Again, for the sake of readability, the subscript i is omitted in the following. Let $\boldsymbol{s} = (r, \dot{r})$ be the state and $u = \ddot{r}$ be the input. The Hamiltonian is then defined by

$$H(\boldsymbol{s}, u, \boldsymbol{\lambda}) = \frac{1}{T} u^2 + \lambda_1 s_2 + \lambda_2 u, \tag{95}$$

where the adjoint variables $\boldsymbol{\lambda}$ must satisfy

$$\dot{\boldsymbol{\lambda}} = -\nabla_{\boldsymbol{s}} H(\boldsymbol{s}, u, \boldsymbol{\lambda}) = (0, -\lambda_1). \tag{96}$$

The result of the adjoint differential equation is

$$\lambda_1(t) = \frac{1}{T} 2d_1, \tag{97}$$

$$\lambda_2(t) = \frac{1}{T}(-2d_1 t - 2d_2). \tag{98}$$

The optimal control input u^* that minimizes the cost function is obtained by

$$u^*(t) = \arg\min_u H(\boldsymbol{s}, u, \boldsymbol{\lambda}), \tag{99}$$

$$= d_1 t + d_2, \tag{100}$$

with the corresponding optimal state trajectories

$$s_1^*(t) = \tfrac{d_1}{6} t^3 + \tfrac{d_2}{2} t^2 + d_3 t + d_4, \tag{101}$$

$$s_2^*(t) = \tfrac{d_1}{2} t^2 + d_2 t + d_3. \tag{102}$$

Since the system dynamics are linear and the cost is convex, it can be shown, analogous to the translational motion primitive, that the control input u^* indeed minimizes the cost (94).

Let $\boldsymbol{R}_0$ and $\boldsymbol{R}_T$ be the initial and final attitude and let $\boldsymbol{\omega}_0$ and $\boldsymbol{\omega}_T$ denote the initial and final angular velocity, respectively. As explained in Section V-B, the motion primitive is always planned from an attitude of $\boldsymbol{r}(0) = 0$ to an attitude of $\boldsymbol{r}(T) = \boldsymbol{r}_e$, where $\boldsymbol{r}_e$ is the rotation between $\boldsymbol{R}_0$ and $\boldsymbol{R}_T$,

$$[\boldsymbol{r}_e] := \log(\boldsymbol{R}_0^T \boldsymbol{R}_T), \tag{103}$$

and afterwards rotated by $\boldsymbol{R}_0$ to satisfy the original attitude constraints. The initial and final states are then given by $\boldsymbol{s}(0) = (0, \omega_0)$ and $\boldsymbol{s}(T) = (r_e, \tilde{\omega}_T)$, where $\tilde{\boldsymbol{\omega}}_T$ is defined to be

$$\tilde{\boldsymbol{\omega}}_T := \boldsymbol{W}^{-1}(\boldsymbol{r}_e) \boldsymbol{\omega}_T. \tag{104}$$

The coefficients $d_1, \ldots, d_4$ can be determined by evaluating the optimal state trajectory $\boldsymbol{s}^*(t)$ at time $t = 0$,

$$\begin{bmatrix} d_3 \\ d_4 \end{bmatrix} = \begin{bmatrix} \omega_0 \\ 0 \end{bmatrix}, \tag{105}$$

and at time $t = T$,

$$\begin{bmatrix} d_1 \\ d_2 \end{bmatrix} = \frac{1}{T^3} \begin{bmatrix} -12 & 6T \\ 6T & -2T^2 \end{bmatrix} \begin{bmatrix} r_e - \omega_0 T \\ \tilde{\omega}_T - \omega_0 \end{bmatrix}. \tag{106}$$

Because the motion primitive is always planned from an initial attitude of $\boldsymbol{r}(0) = 0$, the coefficient d_4 is always zero and will be omitted in the remainder of this section. Finally, combining the optimal state trajectories (101) and (102) along each axis and inserting it into (3) and (9) yields the rotational motion primitive's attitude and angular velocity trajectory (26) and (27).

Analogous to the translational motion primitive, if any component of the initial or final state is subject to optimization, the coefficients d_1, d_2 and d_3 can be determined by evaluating the adapted boundary conditions of the motion primitive, and is done in the following for the different boundary conditions used in this paper. In any case, the trajectories (26) and (27) and cost (33) as introduced in Section V-B remain valid.

State-To-State with Free Initial Angular Velocity:

$$d_2 = 0, \tag{107}$$

$$\begin{bmatrix} d_1 \\ d_3 \end{bmatrix} = \frac{1}{2T^3} \begin{bmatrix} -6 & 6T \\ 3T^2 & -T^3 \end{bmatrix} \begin{bmatrix} r_e \\ \tilde{\omega}_T \end{bmatrix}. \tag{108}$$

State-To-Rest ($\tilde{\omega}_T = 0$) with Free Final Attitude:

$$d_3 = \omega_0, \tag{109}$$

$$\begin{bmatrix} d_1 \\ d_2 \end{bmatrix} = \begin{bmatrix} 0 \\ -\frac{\omega_0}{T} \end{bmatrix}. \tag{110}$$

APPENDIX C
DERIVATION OF MINIMUM-VOLUME BOUNDING BALLS

In this appendix, the minimum-volume bounding balls on a rotation and angular velocity map for a given maximum rotation angle are derived. The findings are used to approximate the thrust and angular velocity constraints during the verification of the motion primitives' feasibility in Section VI.

A. Bounding Ball on Rotation

Consider a vector $\boldsymbol{y} \in \mathbb{R}^3$ and let $\boldsymbol{y}'$ be the result of rotating $\boldsymbol{y}$ about any rotation axis $\boldsymbol{n} \in \mathbb{S}^2$ by a rotation angle $\varphi \in [0, \varphi_{\max}]$, i.e.

$$\boldsymbol{y}' = \boldsymbol{e}^{[\varphi \boldsymbol{n}]}\boldsymbol{y}, \tag{111}$$
$$= \left(\boldsymbol{I} + \sin\varphi[\boldsymbol{n}] + (1-\cos\varphi)\,[\boldsymbol{n}]^2\right)\boldsymbol{y}. \tag{112}$$

Claim: $B(\delta_{\boldsymbol{f}}\boldsymbol{y}, \rho_{\boldsymbol{f}}\|\boldsymbol{y}\|)$ is the minimum-volume bounding ball enclosing all $\boldsymbol{y}'$, with

$$\begin{aligned} \delta_{\boldsymbol{f}} &= \cos\tilde{\varphi}, \\ \rho_{\boldsymbol{f}} &= \sin\tilde{\varphi}, \\ \tilde{\varphi} &= \min\left[\varphi_{\max}, \frac{\pi}{2}\right]. \end{aligned} \tag{113}$$

Proof: Note that independent of the rotation axis, the two terms $[\boldsymbol{n}]\boldsymbol{y}$ and $[\boldsymbol{n}]^2\boldsymbol{y}$ in (112) are perpendicular to each other. For an angle $\alpha \in [0, \pi]$ between $\boldsymbol{n}$ and $\boldsymbol{y}$, the latter term can be expressed by a component parallel and a component perpendicular to $\boldsymbol{y}$, respectively,

$$[\boldsymbol{n}]^2\boldsymbol{y} = \left(\boldsymbol{n}\boldsymbol{n}^T - \boldsymbol{I}\right)\boldsymbol{y}, \tag{114}$$
$$= \cos(\alpha)\|\boldsymbol{y}\|\boldsymbol{n} - \boldsymbol{y}, \tag{115}$$
$$= \cos(\alpha)\|\boldsymbol{y}\|\left(\cos(\alpha)\boldsymbol{n}_{\parallel} + \sin(\alpha)\boldsymbol{n}_{\perp}\right) - \boldsymbol{y}, \tag{116}$$
$$= -\sin(\alpha)^2\|\boldsymbol{y}\|\boldsymbol{n}_{\parallel} + \sin(\alpha)\cos(\alpha)\|\boldsymbol{y}\|\boldsymbol{n}_{\perp}, \tag{117}$$

where

$$\boldsymbol{n}_{\parallel} = \frac{\boldsymbol{y}}{\|\boldsymbol{y}\|}, \text{ and } \boldsymbol{n}_{\perp} = \frac{\boldsymbol{n} - \boldsymbol{n}_{\parallel}\cos(\alpha)}{\|\boldsymbol{n} - \boldsymbol{n}_{\parallel}\cos(\alpha)\|}. \tag{118}$$

Furthermore, it holds that $\|[\boldsymbol{n}]\boldsymbol{y}\| = \sin(\alpha)\|\boldsymbol{y}\|$.

First only rotations with a maximum rotation angle of $\varphi_{\max} \in [0, \frac{\pi}{2}]$, are analyzed. In this case, the maximum distance of any $\boldsymbol{y}'$ from the ball's center is equal or smaller than the ball's radius,

$$\begin{aligned} &\left\|\left(\boldsymbol{I} + \sin(\varphi)[\boldsymbol{n}] + (1-\cos(\varphi))\,[\boldsymbol{n}]^2\right)\boldsymbol{y} - \cos(\varphi_{\max})\boldsymbol{y}\right\|^2 \\ &= \left\|(1-\cos(\varphi_{\max}))\boldsymbol{y} + (1-\cos(\varphi))[\boldsymbol{n}]^2\boldsymbol{y} + \sin(\varphi)[\boldsymbol{n}]\boldsymbol{y}\right\|^2 \\ &= \left((1-\cos(\varphi_{\max})) - (1-\cos(\varphi))\sin(\alpha)^2\right)^2\|\boldsymbol{y}\|^2 \\ &\quad + \left((1-\cos(\varphi))\sin(\alpha)\cos(\alpha)\right)^2\|\boldsymbol{y}\|^2 + \left(\sin(\varphi)\sin(\alpha)\right)^2\|\boldsymbol{y}\|^2 \\ &= (1-\cos(\varphi_{\max}))^2\|\boldsymbol{y}\|^2 \\ &\quad - 2(1-\cos(\varphi_{\max}))(1-\cos(\varphi))\sin(\alpha)^2\|\boldsymbol{y}\|^2 \\ &\quad + (1-\cos(\varphi))^2\sin(\alpha)^2\left(\sin(\alpha)^2 + \cos(\alpha)^2\right)\|\boldsymbol{y}\|^2 \\ &\quad + \sin^2(\varphi)\sin(\alpha)^2\|\boldsymbol{y}\|^2 \\ &= (1-\cos(\varphi_{\max}))^2\|\boldsymbol{y}\|^2 + 2\cos(\varphi_{\max})(1-\cos(\varphi))\sin(\alpha)^2\|\boldsymbol{y}\|^2 \\ &\leq \left((1-\cos(\varphi_{\max}))^2 + 2\cos(\varphi_{\max})(1-\cos(\varphi))\right)\|\boldsymbol{y}\|^2 \\ &\leq \left((1-\cos(\varphi_{\max}))^2 + 2\cos(\varphi_{\max})(1-\cos(\varphi_{\max}))\right)\|\boldsymbol{y}\|^2 \\ &\leq \sin(\varphi_{\max})^2\|\boldsymbol{y}\|^2, \end{aligned} \tag{119}$$

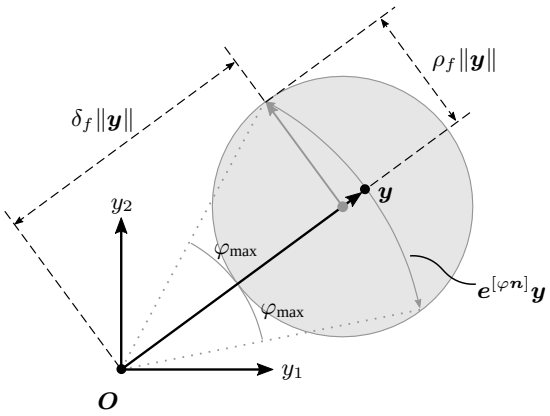


Fig. 10: Illustration of the minimum-volume bounding ball on a rotation with a maximum rotation angle of $\varphi_{\max}$. For the sake of simplicity, only the two-dimensional case is considered with the rotation axis $\boldsymbol{n}$ pointing along the y_3-axis. Any point $\boldsymbol{y}$ that is rotated about the origin $\boldsymbol{O}$ by at most an angle of $\varphi_{\max}$ lies on the circular arc segment with radius $\|\boldsymbol{y}\|$ and subtending angle $2\varphi_{\max}$. The minimum-volume bounding ball enclosing the entire circular arc segment is centered at $\delta_f\boldsymbol{y}$ and has a radius of length $\rho_f\|\boldsymbol{y}\|$.

and the ball therefore encloses all $\boldsymbol{y}'$. Note that (119) holds with equality for any rotation axis perpendicular to $\boldsymbol{y}$ and a rotation angle of $\varphi = \varphi_{\max}$. Therefore, the ball has the minimal radius for the given center. Furthermore, in that case, the ball's radius has the same length as the middle term of (112) that is perpendicular to $\boldsymbol{y}$ and any change of the ball's center would therefore require increasing the ball's radius. Hence, the ball described by (113) is the minimum-volume bounding ball.

If $\varphi_{\max} = \frac{\pi}{2}$, the bounding ball has a radius of $\|\boldsymbol{y}\|$ and is centered at the origin. Therefore, it also encloses all points $\boldsymbol{y}'$ under any rotation with a rotation angle larger than $\frac{\pi}{2}$. ■

A visual interpretation of the minimum-volume bounding ball for a rotation with a rotation angle smaller than $\varphi_{\max}$ is shown in Fig. 10.

B. Bounding Ball on Angular Velocity Map

Let $\boldsymbol{y} \in \mathbb{R}^3$ be any rotation vector velocity and define $\boldsymbol{y}'$ to be the outcome of mapping $\boldsymbol{y}$ to angular velocities under any rotation with rotation axis $\boldsymbol{n} \in \mathbb{S}^2$ and a rotation angle $\varphi \in [0, \varphi_{\max}]$, i.e.

$$\boldsymbol{y}' = \boldsymbol{W}(\varphi\boldsymbol{n})\boldsymbol{y}, \tag{120}$$
$$= \left(\boldsymbol{I} - \frac{1-\cos\varphi}{\varphi}[\boldsymbol{n}] + \frac{\varphi - \sin\varphi}{\varphi}[\boldsymbol{n}]^2\right)\boldsymbol{y}. \tag{121}$$

Claim: $B(\delta_\omega\boldsymbol{y}, \rho_\omega\|\boldsymbol{y}\|)$ is the minimum-volume bounding ball enclosing all $\boldsymbol{y}'$ with

$$\begin{aligned} \delta_\omega &= \begin{cases} 1, & \text{if } \varphi_{\max} = 0, \\ \frac{\sin(\tilde{\varphi})}{\tilde{\varphi}}, & \text{otherwise}, \end{cases} \\ \rho_\omega &= \begin{cases} 0, & \text{if } \varphi_{\max} = 0, \\ \frac{1-\cos(\tilde{\varphi})}{\tilde{\varphi}}, & \text{otherwise}, \end{cases} \\ \tilde{\varphi} &= \min[\varphi_{\max}, \overline{\varphi}], \end{aligned} \tag{122}$$

where $\overline{\varphi}$ is defined the to be the rotation angle at which the ball radius is maximized,

$$\overline{\varphi} := \arg\max_{\varphi>0} \frac{1-\cos(\varphi)}{\varphi}. \tag{123}$$

Proof: Equation (121) is of the same form as (112) and therefore the proof follows the same argumentation as for the minimum-volume bounding ball on rotations.

First, it can easily be verified that (122) is the minimum-volume bounding ball for $\varphi_{\max} = 0$. Next, rotations with a maximum rotation angle of $\varphi_{\max} \in (0, \overline{\varphi}]$ are considered. As in (119), it can be shown that the maximum distance of $\boldsymbol{y}'$ from the ball's center is equal or smaller than the ball's radius, i.e.

$$\begin{aligned}\left\| \left(\boldsymbol{I} - \frac{1-\cos(\varphi)}{\varphi}[\boldsymbol{n}] + \frac{\varphi-\sin(\varphi)}{\varphi}[\boldsymbol{n}]^2 \right) \boldsymbol{y} - \frac{\sin(\varphi_{\max})}{\varphi_{\max}} \boldsymbol{y} \right\|^2 \\ \leq \left(\frac{1-\cos(\varphi_{\max})}{\varphi_{\max}} \right)^2 \|\boldsymbol{y}\|^2,\end{aligned} \tag{124}$$

and that the ball therefore encloses all $\boldsymbol{y}'$. Equation (124) holds with equality for any rotation axis perpendicular to $\boldsymbol{y}$ and a rotation angle of $\varphi = \varphi_{\max}$. Following the same argumentation as for the bounding ball on rotations, any change of the ball's center would require to increase the ball's radius and hence the ball described by (122) is the minimum-volume bounding ball under any rotation with a maximum rotation angle of $\varphi_{\max} = \overline{\varphi}$.

Furthermore, again closely following the proof in (119), it can also be shown that

$$\begin{aligned}\left\| \left(\boldsymbol{I} - \frac{1-\cos(\varphi)}{\varphi}[\boldsymbol{n}] + \frac{\varphi-\sin(\varphi)}{\varphi}[\boldsymbol{n}]^2 \right) \boldsymbol{y} - \frac{\sin(\overline{\varphi})}{\overline{\varphi}} \boldsymbol{y} \right\|^2 \\ \leq \left(\frac{1-\cos(\overline{\varphi})}{\overline{\varphi}} \right)^2 \|\boldsymbol{y}\|^2\end{aligned} \tag{125}$$

holds for any $\varphi > 0$, which implies that all $\boldsymbol{y}'$ under rotations with a rotation angle $\varphi > \overline{\varphi}$ are enclosed by the bounding ball for a maximum angle of $\varphi_{\max} = \overline{\varphi}$. ■

ACKNOWLEDGEMENT

This work is supported by and builds upon prior contributions by numerous collaborators in the Flying Machine Arena. A list of past and present participants of the project is available at `http://flyingmachinearena.org`.

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

REFERENCES

[1] B. Crowther, A. Lanzon, M. Maya-Gonzalez, and D. Langkamp, "Kinematic analysis and control design for a nonplanar multirotor vehicle," *Journal of Guidance, Control, and Dynamics*, vol. 34, no. 4, pp. 1157–1171, 2011.

[2] E. Kaufman, K. Caldwell, D. Lee, and T. Lee, "Design and development of a free-floating hexrotor uav for 6-dof maneuvers," in *2014 IEEE Aerospace Conference*, Mar. 2014, pp. 1–10.

[3] G. Jiang and R. Voyles, "A nonparallel hexrotor uav with faster response to disturbances for precision position keeping," in *2014 IEEE International Symposium on Safety, Security, and Rescue Robotics*, Oct. 2014, pp. 1–5.

[4] M. Ryll, D. Bicego, and A. Franchi, "Modeling and control of fast-hex: A fully-actuated by synchronized-tilting hexarotor," in *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Oct. 2016, pp. 1689–1694.

[5] S. Park, J. Her, J. Kim, and D. Lee, "Design, modeling and control of omni-directional aerial robot," in *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Oct. 2016, pp. 1570–1575.

[6] D. Brescianini and R. D'Andrea, "Design, modeling and control of an omni-directional aerial vehicle," in *2016 IEEE International Conference on Robotics and Automation (ICRA)*, May 2016, pp. 3261–3266.

[7] G. Hoffmann, S. Waslander, and C. Tomlin, "Quadrotor helicopter trajectory tracking control," in *2008 AIAA Guidance, Navigation and Control Conference and Exhibit*, Aug. 2008, pp. 7410–7423.

[8] I. D. Cowling, O. A. Yakimenko, J. F. Whidborne, and A. K. Cooke, "A prototype of an autonomous controller for a quadrotor UAV," in *2007 European Control Conference (ECC)*. IEEE, 2007, pp. 4001–4008.

[9] Y. Bouktir, M. Haddad, and T. Chettibi, "Trajectory planning for a quadrotor helicopter," in *2008 16th Mediterranean Conference on Control and Automation*. IEEE, 2008, pp. 1258–1263.

[10] M. Hehn and R. D'Andrea, "Real-time trajectory generation for quadrocopters," *IEEE Transactions on Robotics*, vol. 31, no. 4, pp. 877–892, Aug. 2015.

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

[12] C. Richter, A. Bry, and N. Roy, "Polynomial trajectory planning for aggressive quadrotor flight in dense indoor environments," in *Robotics Research*. Springer, 2016, pp. 649–666.

[13] M. W. Mueller, M. Hehn, and R. D'Andrea, "A computationally efficient motion primitive for quadrocopter trajectory generation," *IEEE Transactions on Robotics*, vol. 31, no. 6, pp. 1294–1310, Dec. 2015.

[14] M. Hehn, R. Ritz, and R. D'Andrea, "Performance benchmarking of quadrotor systems using time-optimal control," *Autonomous Robots*, vol. 33, no. 1, pp. 69–88, 2012.

[15] M. Geisert and N. Mansard, "Trajectory generation for quadrotor based systems using numerical optimal control," in *2016 IEEE International Conference on Robotics and Automation (ICRA)*, May 2016, pp. 2958–2964.

[16] M. Neunert, C. de Crousaz, F. Furrer, M. Kamel, F. Farshidian, R. Siegwart, and J. Buchli, "Fast nonlinear model predictive control for unified trajectory optimization and tracking," in *2016 IEEE International Conference on Robotics and Automation (ICRA)*, May 2016, pp. 1398–1404.

[17] I. Garcia and J. P. How, "Trajectory optimization for satellite reconfiguration maneuvers with position and attitude constraints," in *2005 American Control Conference (ACC)*. IEEE, 2005, pp. 889–894.

[18] H. Nguyen and Q.-C. Pham, "Time-optimal path parameterization of rigid-body motions: Applications to spacecraft reorientation," *Journal of Guidance, Control, and Dynamics*, vol. 39, no. 7, pp. 1667–1671, 2016.

[19] N. Michael, D. Mellinger, Q. Lindsey, and V. Kumar, "The grasp multiple micro-uav testbed," *IEEE Robotics Automation Magazine*, vol. 17, no. 3, pp. 56–65, Sep. 2010.

[20] 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.

[21] L. E. Kavraki, P. Svestka, J.-C. Latombe, and M. H. Overmars, "Probabilistic roadmaps for path planning in high-dimensional configuration spaces," *IEEE Transactions on Robotics and Automation*, vol. 12, no. 4, pp. 566–580, 1996.

[22] S. M. LaValle and J. James J. Kuffner, "Randomized kinodynamic planning," *The International Journal of Robotics Research*, vol. 20, no. 5, pp. 378–400, 2001.

[23] C. E. Garcia, D. M. Prett, and M. Morari, "Model predictive control: theory and practicea survey," *Automatica*, vol. 25, no. 3, pp. 335–348, 1989.

[24] M. D. Shuster, "A survey of attitude representations," *Navigation*, vol. 8, no. 9, pp. 439–517, 1993.

[25] R. Mahony, V. Kumar, and P. Corke, "Multirotor aerial vehicles: Modeling, estimation, and control of quadrotor," *IEEE Robotics Automation Magazine*, vol. 19, no. 3, pp. 20–32, Sept 2012.

[26] D. P. Bertsekas, *Dynamic Programming and Optimal Control*. Athena Scientific, 2005.

[27] S. Tanygin, "Attitude interpolation," in *AAS/AIAA Spaceflight Mechanics Meeting*, Feb. 2003.

[28] P. Borwein and T. Erdélyi, *Polynomials and polynomial inequalities*. Springer Science & Business Media, 2012, vol. 161.

[29] M. A. Patterson and A. V. Rao, "GPOPS-II: A matlab software for solving multiple-phase optimal control problems using hp-adaptive gaussian quadrature collocation methods and sparse nonlinear programming," *ACM Transactions on Mathematical Software (TOMS)*, vol. 41, no. 1, pp. 1–37, Oct. 2014.
[30] W. C. Durham, "Attainable moments for the constrained control allocation problem," *Journal of Guidance, Control, and Dynamics*, vol. 17, no. 6, pp. 1371–1373, 1994.
[31] A. E. Bryson and Y.-C. Ho, *Applied Optimal Control: Optimization, Estimation and Control*. Ginn and Company, 1969.

Dario Brescianini received the B.Sc. and the M.Sc. degrees in mechanical engineering from ETH Zurich, Zurich, Switzerland, in 2010 and 2013, respectively. He received the Outstanding D-MAVT Bachelor Award and was awarded the Willi-Studer prize for the best Master's degree in Robotics, Systems, and Control.

He is currently a PhD student at the Institute for Dynamic Systems and Control, ETH Zurich. His main research interests include aerial vehicle design, the control and trajectory generation of multirotor vehicles, and learning algorithms.

Raffaello D'Andrea received the B.Sc. degree in Engineering Science from the University of Toronto, Toronto, ON, Canada, in 1991, and the M.S. and Ph.D. degrees in Electrical Engineering from the California Institute of Technology, Pasadena, CA, USA, in 1992 and 1997, respectively.

He was an Assistant and then an Associate Professor with Cornell University, Ithaca, NY, USA, from 1997 to 2007. While on leave from Cornell University, from 2003 to 2007, he cofounded Kiva Systems, North Reading, MA, USA, where he led the systems architecture, robot design, robot navigation and coordination, and control algorithms efforts. He is currently a Professor of Dynamic Systems and Control at ETH Zurich, Zurich, Switzerland, and chairman of the board at Verity Studios AG, Zurich, Switzerland.