A computationally efficient algorithm for state-to-state quadrocopter trajectory generation and feasibility verification

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

Abstract— **An algorithm is proposed allowing for the rapid generation and evaluation of quadrocopter state interception trajectories. These trajectories are from arbitrary initial states to final states defined by the vehicle position, velocity and acceleration with a specified end of time. Sufficient criteria are then derived allowing trajectories to be tested for feasibility with respect to thrust and body rates. It is also shown that the range of a linear combination of the vehicle state can be solved for in closed form, useful e.g. for testing that the position remains within a box. The algorithm is applied by revisiting the problem of finding a trajectory to hit a ball towards a target with a racket attached to a quadrocopter. The trajectory generator is used in a model predictive control like strategy, where thousands of trajectories are generated and evaluated at every controller update step, with the first input of the optimal trajectory being sent to the vehicle. It is shown that the method can generate and evaluate on the order of one million trajectories per second on a standard laptop computer.**

I. Introduction

The popularity of quadrocopters as aerial robotics research platforms has grown rapidly over the past years, with research topics including, for example, vision-based pose estimation [1], nonlinear control [2], cooperative control [3], aerial manipulation [4] and aerial acrobatics [5].

In most applications, it is necessary to plan flight trajectories for quadrocopters in order to describe the planned motion. This problem is complicated by the underactuated and nonlinear nature of the quadrocopter dynamics and large uncertainties caused by aerodynamic effects [6], [7]. Various algorithms have been proposed, focusing on different tradeoffs between computational complexity, high-performance flight, level of detail in which maneuver constraints can be specified, and the ability to handle complex environments.

Broadly speaking, a first group of algorithms handles the trajectory problem by decoupling geometric and temporal planning: in a first step, a geometric trajectory without time information is constructed, for example using lines [8], polynomials [2], or splines [9]. In a second step, the geometric trajectory is parameterised in time in order to guarantee feasibility with respect to the dynamics of quadrocopters.

A second group of algorithms exploits the differential flatness of the quadrocopter dynamics in order to derive constraints on the trajectory, and then solves an optimization problem over a class of trajectories, for example minimum snap [10], minimum time [11], shortest path under uncertain conditions [12], or combinations of position derivatives [13].

The authors are with the Institute for Dynamic Systems and Control, ETH Zurich, Sonneggstrasse 3, 8092 Zurich, Switzerland. {mullerm, hehnm, rdandrea}@ethz.ch

A number of applications require trajectory generation algorithms with particularly short execution times. For example, trajectory generation algorithms can be used as implicit feedback laws similar to model predictive control [11], requiring the computation of a trajectory for each controller update. Higher-level planning algorithms often require the computation of large numbers of trajectories, for example to navigate cluttered environments through randomized target point sampling [13], or to coordinate the collision-free motion of multiple agents [14].

This paper presents a quadrocopter trajectory generation algorithm that is designed to minimize computational complexity. Minimum jerk trajectories are generated without consideration of the dynamic constraints of quadrocopters, and an efficient feasibility verification is carried out a posteriori. The resulting algorithm is capable of generating and verifying trajectories at speeds on the order of one million trajectories per second on a standard laptop computer.

The possibilities offered by a computationally efficient trajectory generation algorithm are demonstrated by its application to the previously presented task of hitting a ball with a racket attached to a quadrocopter [15]. The low computational complexity allows the evaluation of thousands of possible ways to hit the ball at tens of times per second, from which the best way to hit the ball towards a specified target under the given constraints is chosen.

The quadrocopter model is presented in Section II, with the trajectory generation scheme given in Section III. An algorithm to determine feasibility of generated trajectories is introduced in Section IV. The example application of hitting a ball is introduced and results thereof are presented in Section V, and an outlook is given in Section VI.

II. Dynamic model

This section is largely taken from [16], and repeated here for the sake of completeness. The quadrocopter is modelled

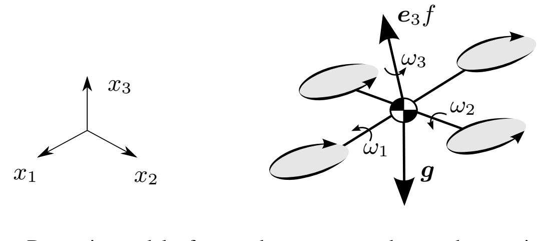

Fig. 1. Dynamic model of a quadrocopter, acted upon by gravity $\boldsymbol{g}$, a thrust force f pointing along the (body-fixed) axis $\boldsymbol{e}_3$; and rotating with angular rate $\boldsymbol{\omega} = (\omega_1, \omega_2, \omega_3)$, with its position in inertial space given as (x_1, x_2, x_3).

as a rigid body with six degrees of freedom: linear translation along the orthogonal inertial x_1, x_2 and x_3 axes, and three degrees of freedom describing the rotation of the frame attached to the body with respect to the inertial frame, which is taken here to be the proper orthogonal matrix $\mathbf{R}$. The control inputs to the system are taken as the total thrust produced f, for simplicity normalised by the vehicle mass and thus having units of acceleration; and the body rates expressed in the body-fixed frame as $\boldsymbol{\omega} = (\omega_1, \omega_2, \omega_3)$, as are illustrated in Fig. 1. It is assumed that high bandwidth controllers onboard the vehicle track the body rate commands, allowing the angular dynamics to be neglected.

The differential equations governing the flight of the quadrocopter are now taken as those of a rigid body [17]

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

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

with $\boldsymbol{g}$ the acceleration due to gravity, $\boldsymbol{e}_3 = (0, 0, 1)$, and $[\![\boldsymbol{\omega}\times]\!]$ the skew-symmetric matrix form of the vector cross product such that

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

A. Feasible inputs

The achievable thrust f produced by the vehicle lies in the range

$$0 < f_{min} \le f \le f_{max} \tag{4}$$

where f_{min} is positive because of the fixed sense of rotation of the propellers. Furthermore, due to saturation limits of the onboard rate gyroscopes, the vehicle's body rates must also lie in the range

$$|\omega_i| \le \omega_{max}, \quad i \in \{1, 2, 3\}. \tag{5}$$

B. Reformulation in jerk

We follow [11] in describing the trajectories of the quadrocopter in terms of the jerk of the axes, allowing the system to be considered as a triple integrator in each axis. It is assumed that a thrice differentiable trajectory $\boldsymbol{x}(t)$ is available, where the jerk is written as $\boldsymbol{j} = \dddot{\boldsymbol{x}} = (\dddot{x}_1, \dddot{x}_2, \dddot{x}_3)$. The input thrust f is found by applying the Euclidean norm $\|\cdot\|$ to (1),

$$f = \|\ddot{\boldsymbol{x}} - \boldsymbol{g}\|. \tag{6}$$

Taking the first derivative of (1) and (6), and simplifying yields respectively

$$\boldsymbol{j} = \mathbf{R}[\![\boldsymbol{\omega}\times]\!]\boldsymbol{e}_3 f + \mathbf{R}\boldsymbol{e}_3\dot{f} \tag{7}$$

$$\dot{f} = \boldsymbol{e}_3^T\mathbf{R}^{-1}\boldsymbol{j}. \tag{8}$$

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

$$\begin{bmatrix} \omega_2 \\ -\omega_1 \\ 0 \end{bmatrix} = \frac{1}{f}\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}\mathbf{R}^{-1}\boldsymbol{j}. \tag{9}$$

Note that ω_3 does not affect the linear motion, and can thus be chosen arbitrarily. A convenient choice is to set $\omega_3 = 0$.

III. Trajectory generation

The problem addressed here is that of finding inputs f and $\boldsymbol{\omega}$ to guide a quadrocopter from an initial state to a final state, each consisting of (at most) a position, velocity and acceleration, in some specified time T. The desired final acceleration can be used to encode a partial final attitude constraint through (1), leaving however one degree of freedom in the final $\mathbf{R}$ consisting of a rotation about $\boldsymbol{e}_3$. This can be understood intuitively by noting that the vehicle's linear dynamics are independent of rotation about $\boldsymbol{e}_3$.

The nonlinear trajectory generation problem is made tractable by decoupling the trajectory into three orthogonal inertial axes, and treating each axis as a triple integrator with jerk as control input. The true control inputs f and $\boldsymbol{\omega}$ are then recovered from the jerk inputs using (6) and (9). Note that feasibility will be neglected until Section IV.

Consider the triple integrator in axis i with the state $s_i = (p_i, v_i, a_i)$, as below, consisting of the scalars position, velocity and acceleration, and the jerk j_i taken as input, such that

$$\dot{s}_i = f(s_i, j_i) = (v_i, a_i, j_i). \tag{10}$$

The goal of the trajectory generator is to guide the system from an initial state to a (possibly only partially defined) final state in a given time T, while minimizing the cost function

$$J_i = \int_0^T j_i(t)^2\,dt \tag{11}$$

where the optimal input and state trajectory are written as j_i^* and s_i^*, respectively. This cost function is chosen primarily for computational convenience, but also has an interpretation as an upper bound on a product of the inputs to the (nonlinear, coupled) quadrocopter system, as described in [16]:

$$\int_0^T f(t)^2\,\|\boldsymbol{\omega}(t)\|^2\,dt \le \sum_{i=1}^{3} J_i. \tag{12}$$

For the sake of readability, the axis subscript i will be discarded for the remainder of this section. The optimal trajectory can be solved straightforwardly by employing Pontryagin's minimum principle (see e.g. [18]) by introducing the costate $\lambda = (\lambda_1, \lambda_2, \lambda_3)$ and defining the Hamiltonian function $H(s, j, \lambda)$:

$$\begin{aligned} H(s, j, \lambda) &= j^2 + \lambda^T f(s, j) \\ &= j^2 + \lambda_1 v + \lambda_2 a + \lambda_3 j \end{aligned} \tag{13}$$

$$\begin{aligned} \dot{\lambda} &= -\nabla_s H(s^*, j^*, \lambda) \\ &= (0, -\lambda_1, -\lambda_2) \end{aligned} \tag{14}$$

The costate differential equation (14) can be easily solved, and for later convenience the solution is written in the constants α, β and γ, such that

$$\lambda(t) = \begin{bmatrix} -2\alpha \\ 2\alpha t + 2\beta \\ -\alpha t^2 - 2\beta t - 2\gamma \end{bmatrix}. \tag{15}$$

The optimal input trajectory is solved for as

$$j^*(t) = \arg\min_j H(s^*(t), j(t), \lambda(t)) \\ = \frac{1}{2}\alpha t^2 + \beta t + \gamma \tag{16}$$

from which the optimal state trajectory follows from (10):

$$s^*(t) = \begin{bmatrix} \frac{\alpha}{120}t^5 + \frac{\beta}{24}t^4 + \frac{\gamma}{6}t^3 + \frac{a_0}{2}t^2 + v_0 t + p_0 \\ \frac{\alpha}{24}t^4 + \frac{\beta}{6}t^3 + \frac{\gamma}{2}t^2 + a_0 t + v_0 \\ \frac{\alpha}{6}t^3 + \frac{\beta}{2}t^2 + \gamma t + a_0 \end{bmatrix} \tag{17}$$

with the initial condition $s(0) = (p_0, v_0, a_0)$.

A. Fully defined end state

Given a fully constrained end state, such that $s(T) = (p_f, v_f, a_f)$, the unknowns α, β and γ are solved by refactoring (17):

$$\begin{bmatrix} \frac{1}{6}T^3 & \frac{1}{2}T^2 & T \\ \frac{1}{24}T^4 & \frac{1}{6}T^3 & \frac{1}{2}T^2 \\ \frac{1}{120}T^5 & \frac{1}{24}T^4 & \frac{1}{6}T^3 \end{bmatrix} \begin{bmatrix} \alpha \\ \beta \\ \gamma \end{bmatrix} = \begin{bmatrix} \Delta a \\ \Delta v \\ \Delta p \end{bmatrix} \tag{18}$$

where

$$\begin{bmatrix} \Delta a \\ \Delta v \\ \Delta p \end{bmatrix} = \begin{bmatrix} a_f - a_0 \\ v_f - v_0 - a_0 T \\ p_f - p_0 - v_0 T - \frac{1}{2}a_0 T^2 \end{bmatrix}. \tag{19}$$

Solving for the unknown coefficients yields

$$\begin{bmatrix} \alpha \\ \beta \\ \gamma \end{bmatrix} = \frac{1}{T^5} \begin{bmatrix} 60T^2 & -360T & 720 \\ -24T^3 & 168T^2 & -360T \\ 3T^4 & -24T^3 & 60T^2 \end{bmatrix} \begin{bmatrix} \Delta a \\ \Delta v \\ \Delta a \end{bmatrix}. \tag{20}$$

B. Partially defined end state

If some components of the final state are left free, the corresponding costates must equal zero at the end time [18]. As an example, if the final velocity is left free, the requirement is that $\lambda_2(T) = 0$. Taking the second component of (15), and the first and third from (17) yields

$$\begin{bmatrix} 2T & 2 & 0 \\ \frac{1}{6}T^3 & \frac{1}{2}T^2 & T \\ \frac{1}{120}T^5 & \frac{1}{24}T^4 & \frac{1}{6}T^3 \end{bmatrix} \begin{bmatrix} \alpha \\ \beta \\ \gamma \end{bmatrix} = \begin{bmatrix} 0 \\ \Delta a \\ \Delta p \end{bmatrix} \tag{21}$$

or

$$\begin{bmatrix} \alpha \\ \beta \\ \gamma \end{bmatrix} = \frac{1}{2T^5} \begin{bmatrix} -15T^2 & 90 \\ 15T^3 & -90T \\ -3T^4 & 30T^2 \end{bmatrix} \begin{bmatrix} \Delta a \\ \Delta p \end{bmatrix} \tag{22}$$

The solutions for the cases of free final acceleration and position, and the various combinations, follow similarly, and are not presented for the sake of brevity.

C. Trajectory cost

The cost value (11) for these trajectories can be solved in closed form as

$$J = \gamma^2 T + \beta\gamma T^2 + \frac{1}{3}\beta^2 T^3 + \frac{1}{3}\alpha\gamma T^3 + \frac{1}{4}\alpha\beta T^4 + \frac{1}{20}\alpha^2 T^5. \tag{23}$$

IV. Determining feasibility

This section will describe computationally inexpensive tests which can be used to verify whether a trajectory generated with the algorithm of Section III is feasible w.r.t. the input constraints, and whether the resulting state trajectories remain within some box constraints.

A. Input feasibility

Given some time interval $\mathcal{T} = [\tau_1, \tau_2]$ and three triple integrator state trajectories of the form (17) with their corresponding jerk inputs j_i, the goal is to determine whether corresponding inputs to the true system f and $\boldsymbol{\omega}$, as used in (1) and (2), satisfy feasibility requirements of Section II-A. The choice of $\mathcal{T}$ will be revisited when describing the recursive implementation. These tests provide sufficient, but not necessary conditions for both feasibility and infeasibility – meaning that some trajectories will be indeterminable w.r.t. these tests – and are designed to be computationally cheap.

1) Thrust: The interval $\mathcal{T}$ is feasible w.r.t. the thrust limits (4) if and only if

$$\max_{t\in\mathcal{T}} f(t)^2 \leq f_{max}^2 \text{ and} \tag{24}$$

$$\min_{t\in\mathcal{T}} f(t)^2 \geq f_{min}^2. \tag{25}$$

Similarly, squaring (6) yields

$$f^2 = \|\ddot{\boldsymbol{x}} - \boldsymbol{g}\|^2 = \sum_{i=1}^{3} (\ddot{x}_i - g_i)^2 \tag{26}$$

where g_i is the component of gravity in axis i. By taking the per-axis extrema of (26) the below bounds follow:

$$\max_{t\in\mathcal{T}} (\ddot{x}_i(t) - g_i)^2 \leq \max_{t\in\mathcal{T}} f(t)^2 \text{ for } i \in \{1,2,3\} \tag{27}$$

$$\max_{t\in\mathcal{T}} f(t)^2 \leq \sum_{i=1}^{3} \max_{t\in\mathcal{T}} (\ddot{x}_i(t) - g_i)^2 \tag{28}$$

$$\min_{t\in\mathcal{T}} f(t)^2 \geq \sum_{i=1}^{3} \min_{t\in\mathcal{T}} (\ddot{x}_i(t) - g_i)^2 \tag{29}$$

Note that the value $\ddot{x}_i - g_i$ as given by (17) is a third order polynomial in time, meaning that its maximum and minimum (denoted $\bar{\ddot{x}}_i$ and $\underline{\ddot{x}}_i$, respectively) can be found in closed form by solving for the roots of a quadratic and evaluating $\ddot{x}_i - g_i$ at at most two points strictly inside $\mathcal{T}$, and at the boundaries of $\mathcal{T}$. The extrema of $(\ddot{x}_i - g_i)^2$ then follow as

$$\max_{t\in\mathcal{T}} (\ddot{x}_i(t) - g_i)^2 = \max\{\bar{\ddot{x}}_i^2, \underline{\ddot{x}}_i^2\} \tag{30}$$

$$\min_{t\in\mathcal{T}} (\ddot{x}_i(t) - g_i)^2 = \begin{cases} \min\{\bar{\ddot{x}}_i^2, \underline{\ddot{x}}_i^2\} & \text{if } \bar{\ddot{x}}_i \cdot \underline{\ddot{x}}_i > 0 \\ 0 & \text{otherwise.} \end{cases} \tag{31}$$

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

Thus, from (27), a sufficient criterion for input infeasibility of the section is if

$$\max\{\bar{\ddot{x}}_i^2, \underline{\ddot{x}}_i^2\} > f_{max}. \tag{32}$$

Similarly from (28)-(29), a sufficient criterion for feasibility is if both

$$\sum_{i=1}^{3} \max\left\{\bar{\ddot{x}}_i^2, \underline{\ddot{x}}_i^2\right\} \leq f_{max} \text{ and} \tag{33}$$

$$\sum_{i=1}^{3} \min\left\{\bar{\ddot{x}}_i^2, \underline{\ddot{x}}_i^2\right\} \geq f_{min} \tag{34}$$

hold. If neither criterion applies, the section is marked as indeterminate w.r.t. thrust feasibility.

2) Body rates: Applying the induced norm to (9) and squaring yields an upper bound for the body rates as a function of the thrust and jerk:

$$\omega_1^2 + \omega_2^2 \leq \frac{1}{f^2}\left\|j\right\|^2 \tag{35}$$

The right hand side of the above can be upper bounded by $\bar{\omega}^2$, defined as below with the denominator evaluated in (31)

$$\bar{\omega}^2 = \frac{\sum_{i=1}^{3} \max_{t\in\mathcal{T}} j_i(t)^2}{\sum_{i=1}^{3} \min_{t\in\mathcal{T}} \left(\ddot{x}_i(t) - g_i\right)^2} \tag{36}$$

This then also provides an upper bound for the sum $\omega_1^2+\omega_2^2$, so that the trajectory section $\mathcal{T}$ can be marked as feasible w.r.t. the body rate input if $\bar{\omega}^2 \leq \omega_{max}^2$, otherwise the section is marked as indeterminate.

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

$$\tau_{\frac{1}{2}} = \frac{\tau_1 + \tau_2}{2} \tag{37}$$

$$\mathcal{T}_1 = [\tau_1, \tau_{\frac{1}{2}}] \tag{38}$$

$$\mathcal{T}_2 = [\tau_{\frac{1}{2}}, \tau_2]. \tag{39}$$

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

Note that as $\tau_{\frac{1}{2}} - \tau_1$ tends to zero the right hand sides of (28)-(29) converge and the thrust feasibility test becomes exact. This does not, however, apply to the body rate feasibility test due to the approximation in (35).

B. Box state constraints

Referring to (17), it can be seen that calculating the range of some linear combination of the system's state can be done by solving for the extrema of a polynomial of order at most five. This involves finding the roots of its derivative, now a polynomial of order at most four, for which closed form solutions exist [19]. This is useful, for example, to verify whether the position of the quadrocopter remains within some bound.

V. Example application

The trajectory generator can be applied to problems involving interception trajectories over a large search space, specifically where the trajectories have clearly defined end states and end times, and the fast execution times allow it to be used in feedback in real-time applications. This is illustrated by revisiting the problem of controlling a quadrocopter with an attached badminton racket tasked with hitting a ball towards a target. This problem was originally investigated in [15], where a near-hover strategy was used to generate reference trajectories without any input/state constraint guarantees.

The experiments were conducted in the Flying Machine Arena, a space equipped with an overhead motion capture system which tracks the pose of the quadrocopters and the position of the balls. The motion capture data is processed by a PC to generate a state estimate, which is then used by the algorithm described in this paper to generate vehicle commands. These commands are transmitted wirelessly to the vehicles at 50 Hz. More information on the system can be found in e.g. [20].

The ball is modelled as a point mass, and impacts between the racket and the ball are modelled as impulses acting purely on the ball, in the direction of the racket normal, with a coefficient of restitution capturing the loss of energy during the collision [15].

There exists a family of candidate flight paths for a ball to move from some starting point to a final point, which will here be indexed by the maximum height h_{max} that the ball achieves – specifically, given some desired h_{max}, and two positions to pass through, the ball flight path is uniquely determined. The pre- and post-impact ball velocities then define the required racket normal direction at impact, and the speed of the racket at impact time [15], while the racket must be at the same position as the ball at impact time. Note

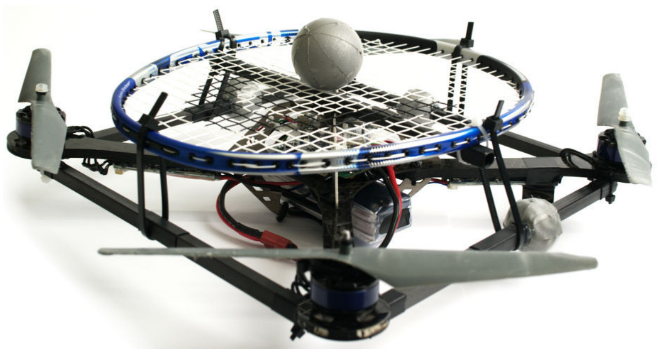

Fig. 2. A quadrocopter with attached badminton racket and a ping-pong ball, as used in the experiments. Note that the racket normal coincides with the vehicle's thrust vector.

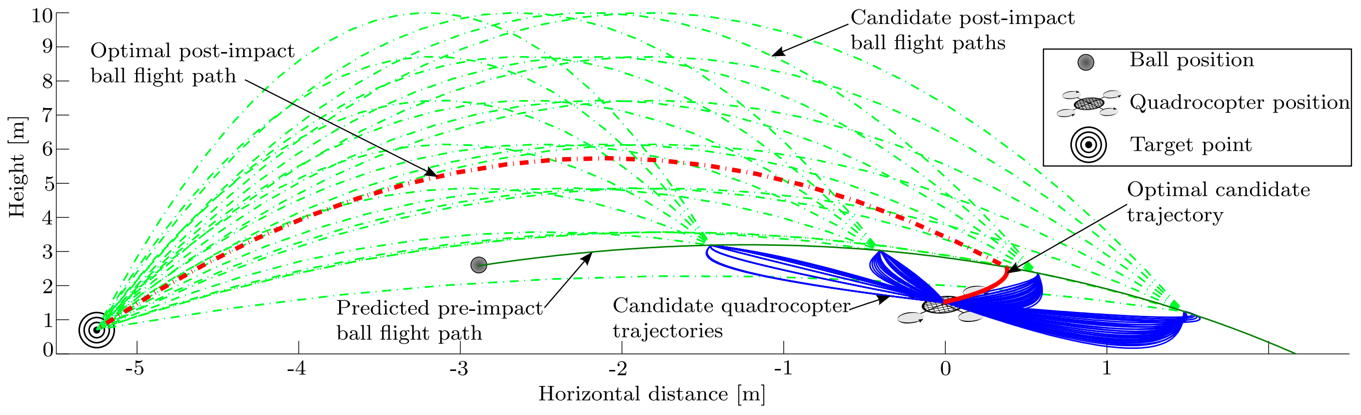

Fig. 3. A snapshot of generated candidate trajectories, showing the current position of the quadrocopter, the ball and the target position. The predicted ball flight path is shown as a solid green line, along which candidate interception points are chosen. For each interception point candidate trajectories are generated ending at one of a set of final thrust values, and hitting the ball through one of a set of maximum heights toward the target, at left in the figure. 100 candidate trajectories are shown of the total of 10'740 candidates created at this time instant. Note that all candidates, including those that are infeasible, are shown. Generating and evaluating the full set of 10'740 candidates took 8.9 ms. The candidate trajectory with the lowest cost is shown in solid bold red, and the first timestep of this trajectory is used to generate an input to send to the quadrocopter. At the next timestep, the ball's predicted flight path and the quadrocopter's current state are updated, and the process is repeated. A detail of four candidates is given in Fig. 4, and the inputs corresponding to the optimal candidate are shown in Fig. 5. A video animation visualising the entire process is available on the first author's web page.

in Fig. 2 that the racket normal is parallel to the vehicle thrust direction $\mathbf{R}\boldsymbol{e}_3$, as defined in (1).

A. Desired quadrocopter end state and trajectory generation

Given an impact location $\boldsymbol{x}_b$ at impact time T, and the maximum height that the ball should pass through to hit its target, the quadrocopter must achieve the following:

1) the same position as the ball: $\boldsymbol{x}(T) = \boldsymbol{x}_b$,
2) a specific racket normal: $\mathbf{R}(T)\boldsymbol{e}_3 = \boldsymbol{n}_R$, and
3) a specific velocity component in the direction of the racket: $\boldsymbol{n}_R \cdot \dot{\boldsymbol{x}}(T) = v_\perp$.

To apply the trajectory generation routine, the second requirement must be transformed to an acceleration value by specifying the thrust value f_f at T, such that $\ddot{\boldsymbol{x}}(T) = \boldsymbol{n}_R f_f + \boldsymbol{g}$. Thus, for every impact location and maximum height, there exist a family of trajectories with different end thrust values f_f that would hit the ball towards the target.

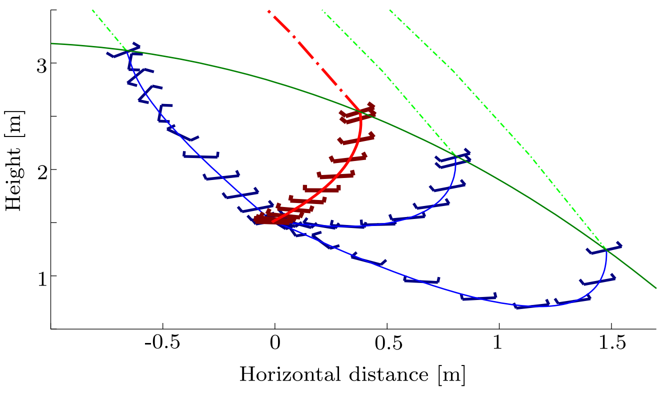

Fig. 4. A detail of four candidate trajectories from Fig. 3, also showing the quadrocopter orientation along the trajectory. The globally optimal trajectory is shown in bold red, with three other candidate trajectories shown in blue. The green line is the ball's predicted pre-impact flight path, while the dash-dotted lines represent the post-impact ball paths. The inputs corresponding to the optimal trajectory are shown in Fig. 5.

The problem is reformulated by introducing a reference frame, related to the inertial reference frame by the constant rotation matrix $\mathbf{R}_b$, such that $\boldsymbol{n}_R = \mathbf{R}_b\,(0,0,1)^1$. The initial and final states are now rotated through $\mathbf{R}_b$ into the reference frame, and (22) is used to generate trajectories in the first two reference directions (with free final velocities), and (20) for the third (fully constrained) reference direction. The trajectory is then tested for feasibility w.r.t. the input constraints (4) and (5), as described in Section IV-A. Next, it is verified whether the quadrocopter position along the trajectory remains within a "safe" box, defined in the original (unrotated) world coordinates – most importantly that the quadrocopter remains above a minimum safe height. This is done in closed form as described in Section IV-B, noting that the rotation through $\mathbf{R}_b$ is a linear operation on the state.

B. Candidate trajectories

At every controller update step, the ball's state is predicted forward until it hits the ground. This predicted trajectory is then discretized in time at 50 Hz to generate a set $\mathcal{S}_T$ of possible ball interception times, where each time has an associated interception point in space. Typically $\mathcal{S}_T$ contains between 15 and 35 elements. A set $\mathcal{S}_h$ of maximum ball heights is also defined, ranging from 1 m to 10 m with 20 equally spaced elements. Finally a set $\mathcal{S}_f$ of end thrust values is defined ranging from $5\,\mathrm{m\,s^{-2}}$ to $20\,\mathrm{m\,s^{-2}}$, also with 20 equally spaced elements.

Typically there are thus thousands of candidate trajectories to generate and evaluate in $\mathcal{S}_T \times \mathcal{S}_h \times \mathcal{S}_f$ at every controller update step, with any self-contradictory elements removed before candidate generation (such as if the desired maximum ball height is below either the impact point or the target point). A cost value defined as the average of the jerk input

[1]Note that this constraint leaves one degree of freedom in $\mathbf{R}_b$, which can be chosen arbitrarily and does not influence the resulting trajectory.

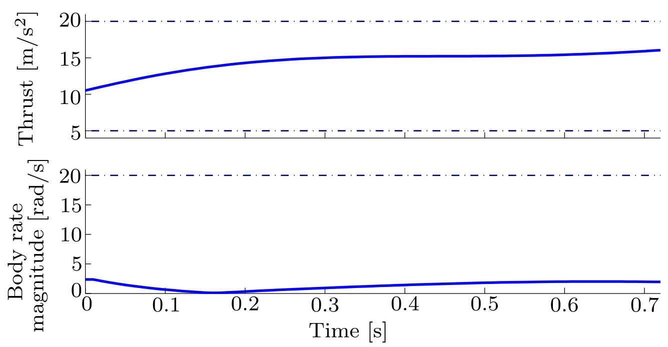

Fig. 5. The inputs corresponding to the optimal candidate trajectory shown in Fig. 3, with the feasible input bounds shown as broken lines. The controller would take the first input step as commands for the quadrocopter.

squared over all axes is assigned to each candidate, defined as

$$J_{\text{candidate}} = \frac{1}{T}\sum_{i=1}^{3} J_i \tag{40}$$

which can be solved for in closed form as given in (23). Note that this cost is invariant under the rotation $\mathbf{R}_b$. Candidates infeasible w.r.t. the input or whose position leaves the safe box are rejected. Furthermore, if a feasible candidate has been found, subsequent candidates with higher cost values can be immediately rejected.

The candidate with the lowest cost value is then used to generate inputs to the system as in model predictive control [21], with the thrust and body rates calculated with (6) and (9). If there are no feasible candidates found, but a previous timestep had yielded a feasible trajectory, the old trajectory is used as reference tracked by a feedback controller. If no candidate is feasible, and no reference trajectory exists, no action is taken and the quadrocopter remains in hover. Snapshots of controller update steps are shown in Fig. 3, and a video animation of the process is available on the first author's web page. The inputs corresponding to the optimal candidate, of which the first timestep input is used as input to the quadrocopter, are shown in Fig. 5.

C. Applied limits

The allowable thrust for the vehicle was limited to $f_{min} = 5\,\text{m}\,\text{s}^{-2}$ and $f_{max} = 20\,\text{m}\,\text{s}^{-2}$, and the allowable body rates to $\omega_{max} = 20\,\text{rad}\,\text{s}^{-1}$. The minimum time section for the input feasibility test was set to $\underline{\Delta\tau} = 20\,\text{ms}$. Finally, the allowable position of the quadrocopter was restricted to lie between 1 and $5\,\text{m}$ above the ground, and to remain within $1.5\,\text{m}$ from the vehicle's start position in either of the horizontal directions.

D. Computation times

The algorithms were implemented on a laptop computer with an Intel Core i7-2620M CPU running at 2.7GHz, with 8GB of RAM, with the solver compiled with Microsoft Visual C++ 11.0. The computation times of the algorithm are investigated by looking at four snapshots similar to Fig. 3. In total, 881'600 candidates were generated, covering four

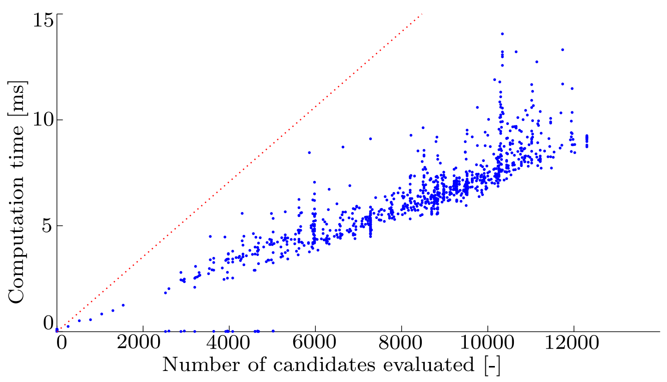

Fig. 6. Time required to generate a control input vs the number of candidates considered. Shown are 1151 controller update steps, taken over a total of 23 ball interception manoeuvres, for a total of 8'187'980 candidates. The dotted line represents the average time cost of evaluating every trajectory for both input feasibility and state box constraints, calculated as 1.77 µs in Section V-D. The actual implementation usually performs better because trajectories that fail one test are immediately discarded.

different ball interceptions (note that here more candidates were considered per intercept to allow for more accurate estimation of the calculation time).

The average time needed to compute the candidate trajectories, and run only the input feasibility test of Section IV-A was $1.21\,\mu\text{s}$. This can be compared to a naïve brute force algorithm, which calculates the actual inputs of the candidate trajectories at times discretized at $50\,\text{Hz}$ and terminates at the first timestep with an infeasible input, which requires on average $10.41\,\mu\text{s}$ per candidate. However, the brute force method is able to correctly distinguish a total of 15'091 (or 1.7% of the total number of trajectories) of additional feasible trajectories (these are trajectories for which the method of Section IV-A terminated with the result 'indeterminable' due to the body rate constraint). The average time per candidate to generate trajectory, and check for both input feasibility as in Section IV-A and position box constraints with the method of Section IV-B was $1.77\,\mu\text{s}$.

Note that the real-time implementation of the trajectories for feedback control as described in Section V-B allows for a lower per-candidate computational time of $0.77\,\mu\text{s}$ on average, as not all tests need to be run on all candidates (since candidates failing one test can be rejected immediately). The time required per controller update step as a function of the number of candidates calculated is shown in Fig. 6.

VI. Conclusion and Outlook

Presented in this paper was a computationally efficient method of generating a trajectory guiding a quadrocopter from an initial state to a final state consisting of position, velocity and/or acceleration in a given time; where the acceleration can be related to two components of the final attitude. The trajectories are optimal in the sense of minimizing an upper bound of the inputs to the quadrocopter. Efficient tests are then presented to determine whether a trajectory is feasible w.r.t. input constraints. It is also shown that the range of the quadrocopter states can be solved for in closed form. The time to calculate a trajectory, and apply the tests, is shown to be on the order of microseconds on a standard laptop PC.

The algorithm is applied to the problem of finding a trajectory to hit a ball towards a target, where it is applied as an implicit feedback controller, similar to model predictive control. In the application, on the order of 10'000 candidate trajectories are calculated and evaluated per controller update step, where the controller is run at 50 Hz.

An interesting extension of the algorithm presented herein would be to provide a bound for the minimum recursion time interval size as a function of tolerances on the input bounds. This could, for example, for a given input tolerance allow for some trajectories to be rejected earlier and at larger interval sizes, speeding up execution.

The algorithm also appears well-suited to other problems requiring the generation of many candidate trajectories, such as probabilistic planning algorithms [22], or the problem of planning for dynamic tasks with multiple vehicles in real time, similar to [23]. The authors intend to investigate this in further work.

ACKNOWLEDGEMENT

The Flying Machine Arena is the result of contributions of many people, a full list of which can be found at `http://www.idsc.ethz.ch/Research_DAndrea/FMA/participants`.

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

REFERENCES

[1] S. Weiss, M. Achtelik, M. Chli, and R. Siegwart, "Versatile distributed pose estimation and sensor self-calibration for an autonomous MAV," in *IEEE International Conference on Robotics and Automation*, 2012, pp. 31–38.

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

[3] D. Mellinger, M. Shomin, N. Michael, and V. Kumar, "Cooperative grasping and transport using multiple quadrotors," in *Proceedings of the International Symposium on Distributed Autonomous Robotic Systems*, Nov 2010, pp. 545–558.

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

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

[6] M. Bangura and R. Mahony, "Nonlinear Dynamic Modeling for High Performance Control of a Quadrotor," in *Australasian Conference on Robotics and Automation*, 2012, pp. 1–10.

[7] G. M. Hoffmann, H. Huang, S. L. Waslander, and C. J. Tomlin, "Precision flight control for a multi-vehicle quadrotor helicopter testbed," *Control engineering practice*, vol. 19, pp. 1023–1036, June 2011.

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

[9] Y. Bouktir, M. Haddad, and T. Chettibi, "Trajectory Planning for a Quadrotor Helicopter," in *Mediterranean Conference on Control and Automation*, Jun. 2008, pp. 1258–1263.

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

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

[12] M. P. Vitus, W. Zhang, and C. J. Tomlin, "A hierarchical method for stochastic motion planning in uncertain environments," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, Villamoura, Portugal, 2012, pp. 2263–2268.

[13] C. Richter, A. Bry, and N. Roy, "Polynomial Trajectory Planning for Quadrotor Flight," in *International Conference on Robotics and Automation*, 2013.

[14] O. Purwin, R. DAndrea, and J.-W. Lee, "Theory and implementation of path planning by negotiation for decentralized agents," *Robotics and Autonomous Systems*, vol. 56, no. 5, pp. 422 – 436, 2008.

[15] M. W. Muller, S. Lupashin, and R. D'Andrea, "Quadrocopter ball juggling," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2011, pp. 5113–5120.

[16] M. W. Mueller and R. D'Andrea, "A model predictive controller for quadrocopter state interception," in *European Control Conference*, 2013, pp. 1383–1389.

[17] P. H. Zipfel, *Modeling and Simulation of Aerospace Vehicle Dynamics Second Edition*. AIAA, 2007.

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

[19] P. Borwein and T. Erdélyi, *Polynomials and Polynomial Inequalities*, ser. Graduate Texts in Mathematics Series. Springer, 1995.

[20] S. Lupashin, A. Schöllig, M. Sherback, and R. D'Andrea, "A simple learning strategy for high-speed quadrocopter multi-flips," in *IEEE International Conference on Robotics and Automation*, 2010, pp. 1642 –1648.

[21] C. E. García, D. M. Prett, and M. Morari, "Model predictive control: Theory and practice–a survey," *Automatica*, vol. 25, no. 3, pp. 335 – 348, 1989.

[22] E. Frazzoli, M. Dahleh, and E. Feron, "Real-time motion planning for agile autonomous vehicles," *Journal of Guidance Control and Dynamics*, vol. 25, no. 1, pp. 116–129, 2002.

[23] M. Sherback, O. Purwin, and R. DAndrea, "Real-time motion planning and control in the 2005 cornell robocup system," in *Robot Motion and Control*, ser. Lecture Notes in Control and Information Sciences, K. Kozlowski, Ed. Springer London, 2006, vol. 335, pp. 245–263.