2013 European Control Conference (ECC)
July 17-19, 2013, Zürich, Switzerland.

A model predictive controller for quadrocopter state interception

Mark W. Mueller and Raffaello D'Andrea

Abstract— **This paper presents a method for generating quadrocopter trajectories in real time, from some initial state to a final state defined by position, velocity and acceleration in a specified amount of time. The end state captures the attitude to within a rotation about the thrust axis. Trajectory generation is done by formulating the trajectory of the quadrocopter in its jerk, in discrete time, and then solving a convex optimisation problem on each decoupled axis. Convex bounds are derived to include feasibility constraints with respect to the quadrocopter's total allowable thrust and angular rates.**

I. Introduction

Quadrocopters are an active area of research, owing to their agility, mechanical simplicity and robustness. They are capable of diverse tasks, and have been used as platforms to study vision-based pose estimation [1], nonlinear control [2], and learning [3], for example. Furthermore, they are useful as tools for solving practical problems such as surveillance [4] and inspection [5].

Various trajectory generation strategies have been proposed, depending on the goal to be achieved. Trajectory generation is complicated by the underactuated and nonlinear nature of the quadrocopter dynamics, and by difficult-to-model aerodynamics (see, e.g. [6], [7] for discussions on aerodynamic effects and [3] for a learning-based compensation strategy).

In [8] a strategy for generating state interception trajectories is presented, where the trajectory is broken down into a sequence of five phases, each part of which is assigned a different controller. These phases consist of: precise hover; 3D path following; attitude control to desired attitude; attitude recovery to zero angles; and finally, soft hover control for recovery.

The strategy presented in [9] involves exploiting the differential flatness of the quadrocopter dynamics, and is used to generate trajectories between so-called "keyframes", which are defined as positions in space and yaw angles. The generated trajectories then pass through keyframes at specified times, while minimizing the snap (fourth derivative of position) squared. These trajectories are solved using either a normalised time, or distance, and are then scaled to the problem at hand.

Pontryagin's minimum principle is exploited in [10] and [11] to generate: 1) bang-singular to-rest quadrocopter trajectories, and 2) bang-singular position interception trajectories, where the goal is to reach a given position at a given time, while minimizing the time required to stop after the intercept. These methods are computationally fast, with solution times on the order of microseconds.

The differential flatness of the quadrocopter dynamics was also exploited in [2] to generate trajectories, where the authors make the simplification that the roll-pitch-yaw Euler angle accelerations are control inputs. The optimisation is done on a weighted sum of the fuel cost (approximated as average speed) and deviation from desired arrival time. Trajectories are then solved for by choosing accelerations as polynomials in time, with the degree of the polynomial being a function of the number of boundary conditions, and then solving for the polynomial coefficients.

In [12] collision-free trajectories are generated to guide a fleet of UAVs from initial states to final states, guaranteeing that the trajectories maintain a minimum distance whilst minimising the total thrust produced by the quadrocopters. The solutions are found using sequential convex programming, with solution times on the order of seconds.

A learning-based model predictive controller (MPC) is presented in [13], with the Euler angles taken as inputs; a cascaded MPC is designed for a quadrocopter in [14], with the dynamics of the quadrocopter captured by piecewise affine equations, separating control of the attitude and planar motions; while MPC is combined with robust control in [15]. In each case, MPC is used to track a given state trajectory.

This paper builds on a previously presented scheme for generating trajectories for generating trajectories for a quadrocopter hitting a ball with an attached racket [16]. That method required that the vehicle maintain small pitch and roll angles, with the end state restricted to the position, one component of the velocity, and the direction of the racket normal.

Here, a scheme is presented for generating state interception trajectories for quadrocopters; that is, trajectories starting from an arbitrary state and achieving a (reduced) end state in a specified amount of time, whilst satisfying input constraints. The desired end state is specified as the position, velocity and acceleration of the vehicle, i.e. fixing the attitude of the vehicle to within a rotation about the vehicle's thrust axis. The total thrust and body rates are bounded by convex functions, allowing the problem to be written as a convex constrained optimisation problem, which can be solved in real time on a typical desktop computer, and which allows the trajectory generator to be used as a diminishing horizon model predictive controller.

This method extends the state of the art by calculating state interception trajectories in real time, using sophisticated optimisation techniques to explicitly include input constraints in the trajectory generation problem.

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

The quadrocopter model is presented in Section II, with the trajectory generation scheme given in Section III. Section IV discusses implementation and experimental results, and an outlook is given in Section V.

II. Dynamic model

The quadrocopter is modelled as a rigid body with six degrees of freedom: linear translation along the 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 $\boldsymbol{\omega} = (\omega_1, \omega_2, \omega_3)$. These are illustrated in Fig. 1.

The mixing of these inputs to individual motor thrust commands is done on board the vehicle, using feedback from gyroscopes. It is assumed that the time constant of the onboard controllers is low enough to have negligible influence on the algorithm presented here. Because of their low rotational inertia, quadrocopters can achieve extremely high rotational accelerations (on the order of $200\,\mathrm{rad\,s^{-2}}$ [10]) about the ω_1 and ω_2 axes, while it will be shown that the rotation about ω_3 is not needed for the trajectories considered here. For example, [17] presents a scheme for mixing these inputs to motor commands.

The differential equations governing the flight of the quadrocopter are now taken as those of a rigid body [18]

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

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

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

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

and $\boldsymbol{g} = (0,0,-g)$ the acceleration due to gravity. Note the distinction between the vector $\boldsymbol{g}$ and scalar g.

A. Reformulation in jerk

We follow [10] in considering 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 and simplifying the trajectory generation task.

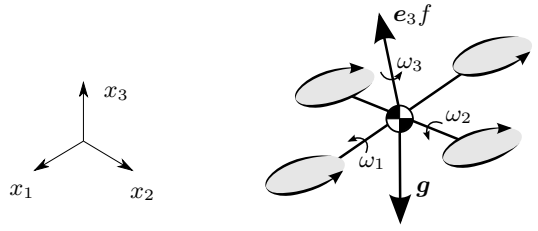

Fig. 1. Dynamic model of a quadrocopter, acted upon by gravity $\boldsymbol{g}$, a thrust force f pointing along the (body-fixed) axis $\boldsymbol{e}_3$; and rotating with angular rate $\boldsymbol{\omega} = (\omega_1, \omega_2, \omega_3)$, with its position in inertial space given as (x_1, x_2, x_3).

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 then found by applying the Euclidean norm $\|\cdot\|$ to (1),

$$f = \|\ddot{\boldsymbol{x}} - \boldsymbol{g}\|\,. \tag{4}$$

Squaring the above, taking the derivative and substituting for (1) yields

$$2f\dot{f} = 2\left(\ddot{\boldsymbol{x}} - \boldsymbol{g}\right)^T \boldsymbol{j} = 2\left(\mathbf{R}\boldsymbol{e}_3 f\right)^T \boldsymbol{j} \tag{5}$$

$$\dot{f} = \boldsymbol{e}_3^T \mathbf{R}^T \boldsymbol{j}. \tag{6}$$

Taking the first derivative of (1) yields

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

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:

$$\begin{bmatrix} \omega_2 \\ -\omega_1 \\ 0 \end{bmatrix} = \frac{1}{f} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} \mathbf{R}^T \boldsymbol{j}. \tag{8}$$

That the third component of body rates, ω_3, does not appear can be understood by noting that a rotation about the $\boldsymbol{e}_3$ axis does not affect the translational acceleration (1).

Using (4) and (8), the system inputs are given for a trajectory described in its jerk, with one remaining degree of freedom in ω_3. This could be fully specified if the full attitude of the quadrocopter were also known (specifically, the rotation about the thrust axis). For simplicity, here it will be assumed that $\omega_3 = 0$ and that this rotation is unimportant.

B. Feasibility constraints and decoupled axes

A quadrocopter trajectory described by (1) and (2) is considered to be feasible if the thrust and the magnitude of the body rates lie in some feasible set, defined as

$$0 < f_{\min} \leq f \leq f_{\max} \tag{9}$$

$$\|\boldsymbol{\omega}\| \leq \omega_{\max}. \tag{10}$$

Note that $f_{\min} > 0$ for fixed-pitch propellers with a fixed direction of rotation, and, specifically, that the requirement on the thrust input is non-convex. These limits are translated to limits on the jerk trajectory by squaring (4) and writing it in its components:

$$f_{\min}^2 \leq \ddot{x}_1^2 + \ddot{x}_2^2 + (\ddot{x}_3 + g)^2 \leq f_{\max}^2. \tag{11}$$

The following conservative box constraints are applied to yield convex constraints:

$$\ddot{x}_{\min\{1\}} = -\ddot{x}_{\max\{1\}} \leq \ddot{x}_1 \leq \ddot{x}_{\max\{1\}} \tag{12}$$

$$\ddot{x}_{\min\{2\}} = -\ddot{x}_{\max\{2\}} \leq \ddot{x}_2 \leq \ddot{x}_{\max\{2\}} \tag{13}$$

$$\ddot{x}_{\min\{3\}} = f_{\min} - g \leq \ddot{x}_3 \leq \ddot{x}_{\max\{3\}}. \tag{14}$$

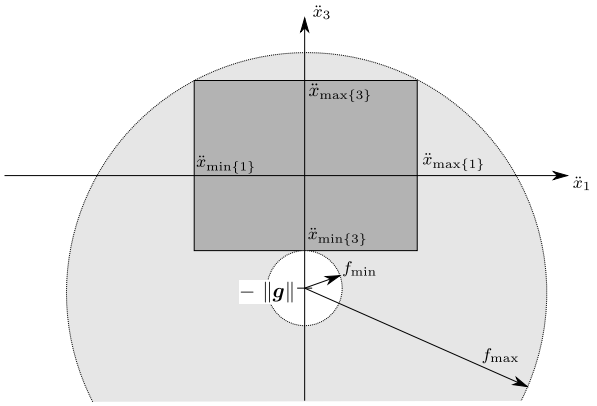

Fig. 2. A cross-section of the feasible acceleration sets for a quadrocopter. The lightly shaded, non-convex, doughnut defines the true thrust limits of the vehicle, while the darker rectangular area defines the decoupled per-axis acceleration limits used here. Note that the circle of radius $f_{\max}$ is truncated in the graphic.

The resulting trajectories are guaranteed to be feasible with respect to the thrust limit if

$$\ddot{x}_{\max\{1\}}^2 + \ddot{x}_{\max\{2\}}^2 + \left(\ddot{x}_{\max\{3\}} + g\right)^2 \leq f_{\max}^2 \tag{15}$$

as visualised for two axes in Fig. 2.

By taking the (induced) norm of (8), an upper bound for the body rates can be found as a function of the jerk, as

$$\|\boldsymbol{\omega}\| \leq \frac{1}{f} \|\boldsymbol{j}\| \leq \frac{1}{f_{\min}} \|\boldsymbol{j}\|. \tag{16}$$

Applying the limit (10) to the above, and rearranging yields and upper bound on the allowable jerk per axis

$$j_{\max} = \frac{1}{\sqrt{3}} f_{\min}\, \omega_{\max} \tag{17}$$

under the worst case that all three axes produce the maximum allowable jerk $j_{\max}$ at the same time as the minimum thrust $f_{\min}$ is achieved.

III. Trajectory generation

Considering the system input to be the three-dimensional jerk, the quadrocopter dynamics become a set of three triple integrators, one in each axis, with states position, velocity and acceleration. The trajectory generation is rewritten as an optimal control problem, with boundary conditions defined by the quadrocopter's initial and (desired) final states. The cost function to minimize is chosen as

$$J_{\text{coupled}} = \int_0^T \left(j_1(t)^2 + j_2(t)^2 + j_3(t)^2\right) dt. \tag{18}$$

Note that, by rearranging (16), this cost function can be interpreted as an upper bound for a product of the inputs, since

$$f^2 \|\boldsymbol{\omega}\|^2 \leq j_1^2 + j_2^2 + j_3^2. \tag{19}$$

This implies that the cost function (18) can be split, such that each axis is minimized independently, while remaining meaningful in the context of the coupled three-dimensional trajectory.

A. Discrete time formulation

The trajectory generation problem for each decoupled axis is rendered finite dimensional by discretizing the time with uniform steps of size Δt. Each axis is then a discrete time linear, time invariant system in the state z, consisting of position, velocity and acceleration, with scalar jerk input $j = \dddot{x}$, where the axis subscripts have been neglected for convenience.

$$j[k] = \dddot{x}(k\Delta t) \tag{20}$$

$$z[k] = \begin{bmatrix} x(k\Delta t) & \dot{x}(k\Delta t) & \ddot{x}(k\Delta t) \end{bmatrix}^T \tag{21}$$

$$z[k+1] = A_d z[k] + B_d j[k] \tag{22}$$

$$A_d = \begin{bmatrix} 1 & \Delta t & \frac{1}{2}\Delta t^2 \\ 0 & 1 & \Delta t \\ 0 & 0 & 1 \end{bmatrix} \tag{23}$$

$$B_d = \begin{bmatrix} \frac{1}{6}\Delta t^3 & \frac{1}{2}\Delta t^2 & \Delta t \end{bmatrix}^T \tag{24}$$

The optimal control problem is to minimize the cost function

$$J = \sum_{k=0}^{N} j[k]^2 \tag{25}$$

subject to the above dynamics, satisfying the boundary conditions defined by the initial and final positions (x_0 and x_f, respectively), velocities ($\dot{x}_0$ and $\dot{x}_f$) and accelerations ($\ddot{x}_0$ and $\ddot{x}_f$):

$$z[0] = \begin{bmatrix} x_0 & \dot{x}_0 & \ddot{x}_0 \end{bmatrix}^T \tag{26}$$

$$z[N] = \begin{bmatrix} x_f & \dot{x}_f & \ddot{x}_f \end{bmatrix}^T \tag{27}$$

with the end stage calculated from the end time T as $N = \text{Round}(T/\Delta t)$.

The constraints on acceleration of (12) - (14) and the jerk limit (17) are affine functions of the state $z[k]$ and input $j[k]$ (where n is the axis under consideration):

$$\begin{bmatrix} 0 & 0 & 1 \\ 0 & 0 & -1 \end{bmatrix} z[k] \leq \begin{bmatrix} \ddot{x}_{\min\{n\}} \\ \ddot{x}_{\max\{n\}} \end{bmatrix} \tag{28}$$

$$\begin{bmatrix} 1 \\ -1 \end{bmatrix} j[k] \leq \begin{bmatrix} j_{\max} \\ -j_{\max} \end{bmatrix}. \tag{29}$$

The quadratic cost function (25) with the linear equality constraints (22), (26) and (27), and the affine inequality constraints of (28) and (29), together define a convex optimisation problem. This is a special case of model predictive control [19] with a fixed end constraint, and with a diminishing rather than receding horizon (i.e. the trajectory is only planned to intercept).

There exist efficient methods for solving problems of this sort, with CVXGen [20], FORCES [21] and FiOrdOs [22] presenting techniques for creating C-code based solvers for specific instances of convex optimization problems.

Here, solvers are generated using the FORCES software of [21], which was able to generate solvers for large problems (here trajectories up to $N = 200$ are considered). FORCES uses efficient interior point methods tailored to convex multistage problems, as are typical in model predictive control

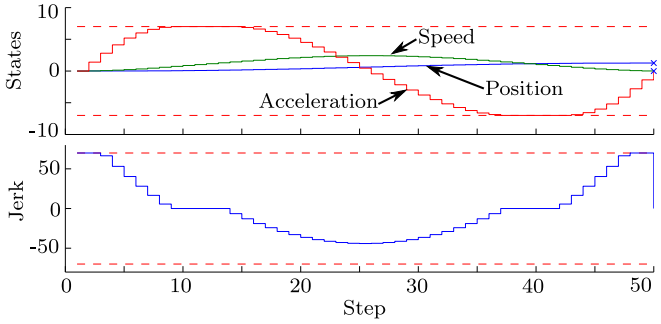

Fig. 3. Example one-dimensional constrained trajectory solution, starting at rest at the origin and ending at $x_f = 1.25\,\mathrm{m}$, $\dot{x}_f = \ddot{x}_f = 0$ in $T = 1\,\mathrm{s}$, with acceleration limits $\ddot{x}_{\max} = -\ddot{x}_{\min} = 7\,\mathrm{m\,s^{-2}}$ and $j_{\max} = 70\,\mathrm{m\,s^{-3}}$. Each step represents 0.02 s.

applications, and allows for high-speed implementation with good numerical stability properties [21].

The generated solvers either return a solution that solves the problem to within some acceptable residuals, or returns that no solution is found. In reality, failure to find a solution can mean that:

- a solution exists, but the solver failed to find it due to reaching an internal limit;
- no solution exists to the conservatively constrained decoupled problem;
- no solution exists to the fully coupled nonlinearly constrained problem.

An example trajectory is shown in Fig. 3, where a one-dimensional trajectory is generated over a translation of 1.25 m from rest at the origin to rest in 1 s (or 50 steps at 50 Hz). The acceleration limits are set to $\ddot{x}_{\max} = -\ddot{x}_{\min} = 7\,\mathrm{m\,s^{-2}}$ and the jerk limit to $j_{\max} = 70\,\mathrm{m\,s^{-3}}$. The minimum and maximum acceleration limits, and the maximum jerk limit, are active for portions of this trajectory.

B. Solution time

The statistical performance of the trajectory generator is investigated by solving trajectories from rest, for interception times varying from 0.2 to 4 s, in 20 ms intervals (i.e. 10 to 200 steps). For each trajectory length, 1000 end states are chosen uniformly at random in the range $x_f \in [-3, 3]\,\mathrm{m}$, $\dot{x}_f \in [-2, 2]\,\mathrm{m\,s^{-1}}$ and $\ddot{x}_f \in [-5, 5]\,\mathrm{m\,s^{-2}}$, and applying the constraints $\ddot{x}_{\max} = -\ddot{x}_{\min} = 7\,\mathrm{m\,s^{-2}}$ and $j_{\max} = 70\,\mathrm{m\,s^{-3}}$. The mean solution time over all 191000 trajectories was 1.5 ms, with a maximum of 13.7 ms.

These were calculated on a PC running Windows 7, with an Intel Core i7-2620M CPU at 2.70 GHz, with 4 GB RAM. The solver was compiled into a dynamically linked library using the Intel C++ Composer XE Windows: 2011.8.278, which was subsequently linked to an executable using Visual Studio 2008. In each case the optimisations were set to maximise speed.

C. Completeness

Because the generated solver does not check for feasibility, it does not distinguish between infeasible trajectories and

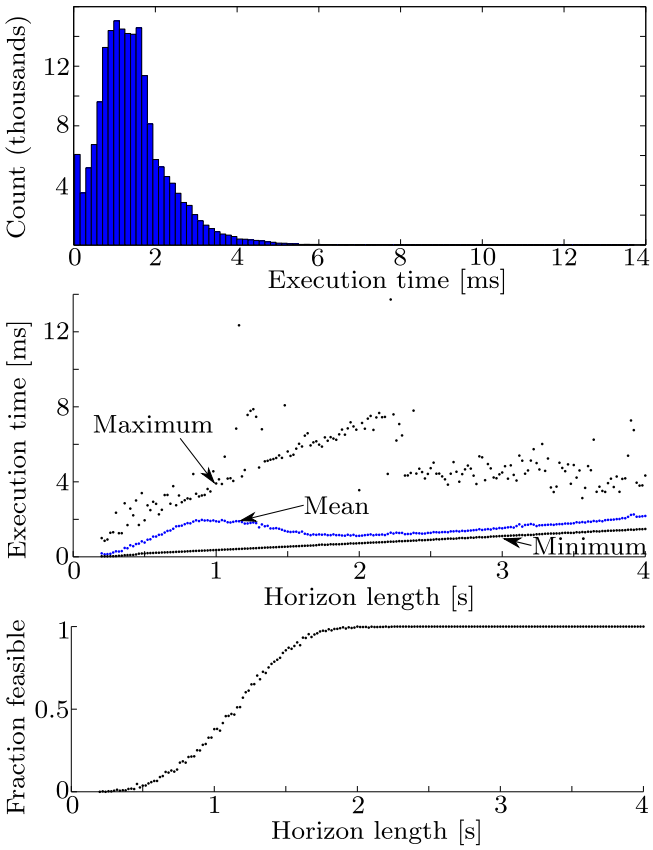

Fig. 4. Performance of the constrained triple integrator trajectory solver, with trajectories generated from rest to a uniformly distributed random final state. The top plot shows a histogram of the solution times, using the data for all trajectory lengths, while the middle plot shows the execution time as a function of horizon length, and the bottom plot shows the fraction of randomly generated endstates for which a trajectory could be found as a function of horizon length. The reduction in mean solution time at $T \approx 1\,\mathrm{s}$ can be explained by noting that, for shorter end times, almost all trajectories failed to find a solution.

those that hit iteration limits of the solver. The frequency of occurrence of these false negatives is investigated by generating one-dimensional trajectories from rest to end states of positions in the range $[0, 3.5]\mathrm{m}$, speeds in the range $[0, 5]\mathrm{m\,s^{-1}}$, and zero acceleration; with an end time of 1 s (50 steps) and limits as in the example trajectory above. The problem was also modelled in Yalmip [23], and solved as a quadratic program, allowing the detection of infeasible problems. Note that completeness here refers only to the decoupled triple integrator, not the fully coupled quadrocopter model; a discussion of the latter can be found in [24].

The results are visualised in Fig. 5: of 10'000 end states evaluated, 5278 were found to be feasible; of these feasible end states, the proposed solver found 85.0%.

IV. Validation

The Flying Machine Arena (FMA) at the ETH Zurich is a platform for design and validation of autonomous aerial systems, and consists of a large motion capture volume and a fleet of quadrocopters. Commands for the three body rates

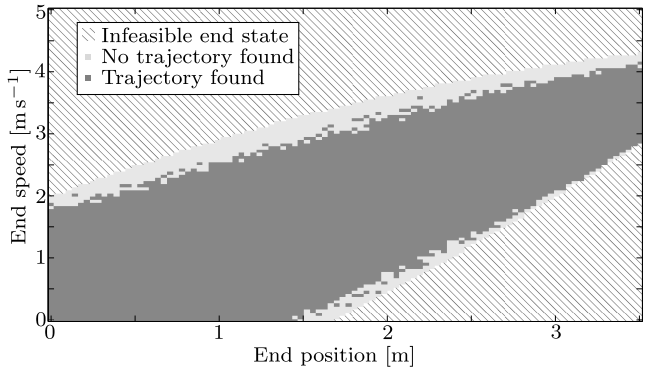

Fig. 5. Completeness of the trajectory solver, showing whether a one-dimensional trajectory was found from rest to a specified end state within 1 s for 10'000 different end states. The middle grey area represents feasible end states for which the solver hit an internal limit, and failed to find a trajectory.

and the collective thrust are sent at 50 Hz over a wireless channel, and an onboard controller uses rate gyro measurements to generate motor thrust commands. The motion capture system is used to measure the position and attitude of the quadrocopter, fused into a full state estimate to be used for control.

The trajectory generator is implemented as a model predictive controller, solving a trajectory to intercept at each time step from the vehicle's current state, and applying the first input to the system. If the trajectory generator fails, the trajectory from the previous time step is followed using a feedback controller. Since the axes are independent, the constrained solution for each axis can be solved for in parallel, implying that a worst-case run-time of 13.7 ms (from the above statistical analysis) would still fit in the 20 ms control period.

The thrust and body rates are limited as below, where these values have been observed to match the FMA vehicles well.

$$f_{\min} = 5\,\mathrm{m\,s^{-2}} \tag{30}$$

$$f_{\max} = 20\,\mathrm{m\,s^{-2}} \tag{31}$$

$$\boldsymbol{\omega}_{\max} = 25\,\mathrm{rad\,s^{-1}} \tag{32}$$

A. Easy trajectory

An "easy" trajectory is examined first, where the goal is to translate 1 m in the x_1 direction, starting at rest and ending at rest. The upper thrust limit is divided amongst all axes equally, and this trajectory is considered "easy" because the planned trajectories at time 0 does not have any active input constraints.

$$\ddot{x}_{\max\{1\}} = \ddot{x}_{\max\{2\}} = \ddot{x}_{\max\{3\}} \approx 7.31\,\mathrm{m\,s^{-2}} \tag{33}$$

The resulting flight is shown in Fig. 6, specifically, the system reached the end state to within state errors of 49 mm in position, $0.10\,\mathrm{m\,s^{-1}}$ in velocity and $1.1\,\mathrm{m\,s^{-2}}$ in acceleration, while remaining inside the feasibility constraints. The figure also shows that, although the initially planned trajectory does not hit any input constraints, later replanning results in trajectories that do have active constraints, due to disturbances.

B. Hard trajectory

Next, a "hard" trajectory is considered, one which involves active input constraints in each axis. As with the easy trajectory, the upper thrust limit was split equally amongst all axes. The quadrocopter starts at rest at the origin, with a target end state characterised by

$$x_f = (3, -3, 2)\,\mathrm{m}, \tag{34}$$

$$\dot{x}_f = (5, 0, 0)\,\mathrm{m\,s^{-1}}, \tag{35}$$

$$\ddot{x}_f = (0, 4.9, 0)\,\mathrm{m\,s^{-2}}; \tag{36}$$

in a time of $T = 1.5\,\mathrm{s}$. These end constraints can be interpreted as trying to pass through a window approximately 4.69 m away, with a speed of $5\,\mathrm{m\,s^{-1}}$ at an attitude at 30° from the horizontal with no vertical acceleration at the end.

A resulting trajectory is shown in Fig. 7, with the achieved end state

$$x_f = (2.89, -2.75, 1.89)\,\mathrm{m}, \tag{37}$$

$$\dot{x}_f = (4.65, 0.28, 0.44)\,\mathrm{m\,s^{-1}}, \tag{38}$$

$$\ddot{x}_f = (-0.49, 5.96, 0.78)\,\mathrm{m\,s^{-2}}; \tag{39}$$

or a position error of 0.29 m, velocity error of $0.63\,\mathrm{m\,s^{-1}}$. The direction of thrust at the end differs from the desired by an angle of 3.6°. The position error equates to approximately 6% of the total translation, and the velocity error to 13% of the total velocity change. From approximately 0.6 s onwards, no feasible trajectories can be found from the vehicle's current position, and the last valid trajectory is flown in feedback. This is possible because the constraints used for trajectory generation are conservative, especially as the true feasible regions are non-convex.

The inputs applied during this trajectory are shown in Fig. 8, where it can be seen that the inputs stay within the thrust and body rate constraints. The individual axes do briefly violate the acceleration constraints during the period where no feasible trajectory could be found from the current position (therefore the vehicle was flying on feedback with the last valid trajectory as reference). The conservative nature of the body rate limits of (17) can be also seen: the commanded body rates are generally very low, rising only under trajectory following feedback control near intercept.

C. Box constraint selection

If additional information on the trajectory is available in advance, or sufficient planning time is available, the convex bounds on the accelerations and jerk can be chosen more efficiently (refer to Fig. 2). Similarly, if the trajectory is solved for off line, the bounds can be iterated on to make use of them more efficiently.

As an example, consider a rest-to-rest motion, translating 4 m along the x_1 axis. Using the general limits as above, the fastest end time for which a feasible trajectory is found is 1.60 s. However, because the motion is only along one axis, the box constraints on the acceleration of the other two axes can be tightened. Specifically, limiting the accelerations in the x_2 and x_3 axes to $\pm 1\,\mathrm{m\,s^{-2}}$ (to maintain the ability

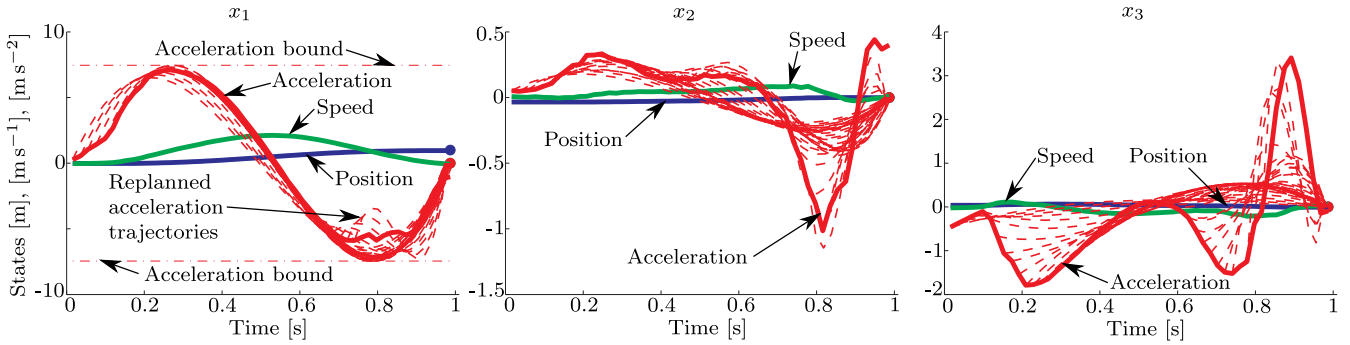

Fig. 6. Resulting flight for the "easy" trajectory; refer to Section IV-A, where the goal is a rest-to-rest translation of 1m in x_1. The plot shows the trajectory for each axis (from left to right: x_1, x_2 and x_3) separately. The solid line shows the actual flown trajectory, and the dashed lines indicate the planned acceleration trajectories, as replanned at each stage. Note that, from approximately 0.8s onwards, feasible trajectories can no longer be found and the vehicle flies in feedback on the last valid trajectory.

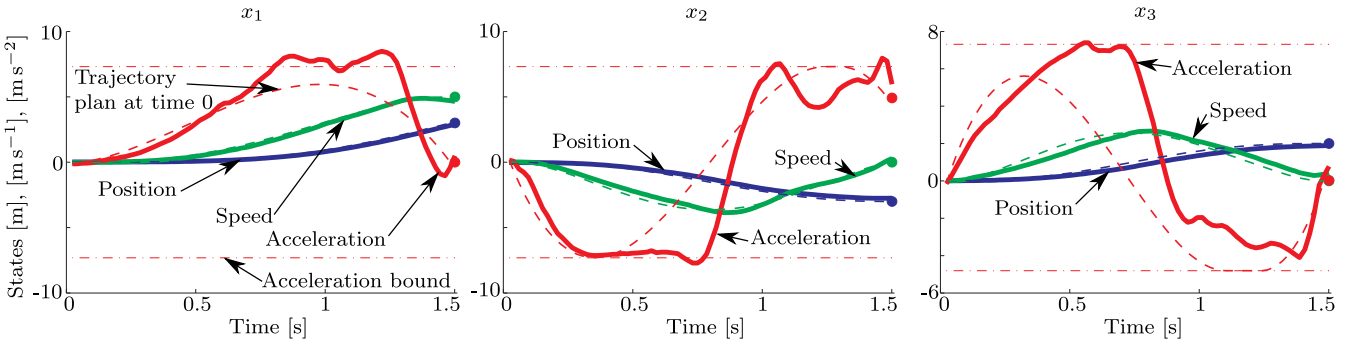

Fig. 7. Resulting trajectories for the "hard" trajectory of Section IV-B: a quadrocopter starting at rest at the origin, and flying to an end state of $x_f = (3, -3, 2)\,\mathrm{m}$, $\dot{x}_f = (5, 0, 0)\,\mathrm{m\,s^{-1}}$ and $\ddot{x}_f = (0, 4.9, 0)\,\mathrm{m\,s^{-2}}$ in $T = 1.5\,\mathrm{s}$. Each plot shows the trajectory along a different axis in inertial space. The thick lines represent the actual trajectory as followed by the quadrocopter, while the thin broken line is the solution to the trajectory generation problem as obtained at the first time instant. Up to approximately 0.6 s, a new trajectory is generated at each time step (not shown), but from then onwards feasible trajectories can no longer be found and feedback is done using the last valid trajectory as reference. Refer to Fig. 8 for the corresponding inputs.

to compensate for disturbances), the limits for x_1 can be raised to $\pm 16.80\,\mathrm{m\,s^{-2}}$. Furthermore, the minimum thrust value can be raised to $8.81\,\mathrm{m\,s^{-2}}$, allowing the jerk limit to be raised to $127.16\,\mathrm{m\,s^{-3}}$ by (17). Using these limits, the trajectory generator finds a solution for an end time of 1.24 s, an improvement of 22.5%. Note that these bounds are still conservative, due to the non-convex nature of the true feasible region.

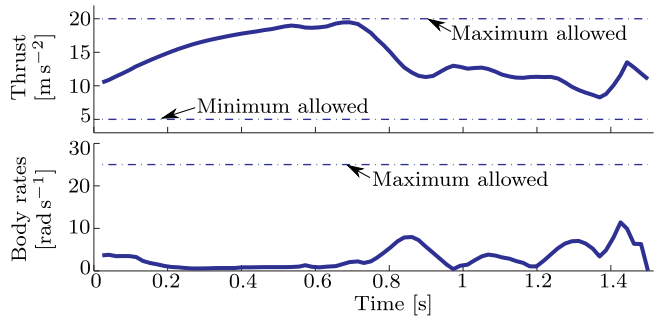

Fig. 8. Inputs for the hard trajectory of Section IV-B. Note that the thrust limits are hit, but the magnitude of the body rate command stays well below the maximum allowed. This due to the conservative nature of the limit (17).

V. Outlook

This paper presents a method for calculating state interception trajectories for quadrocopters, from arbitrary initial conditions to a desired end state characterised by a position, velocity and acceleration, resulting in commands of thrust and two components of the body rates. The end acceleration constraint can be viewed as defining the attitude of the vehicle to within a rotation about the thrust axis. A possible extension to this research would be calculating an input trajectory for the third component of the body rates, such that the final degree of freedom of the attitude can also be specified.

The constrained trajectory generation problem is posed as a convex optimisation problem for each decoupled axis, which can be solved in real time. Therefore this technique is suitable for use in feedback as a model predictive controller; it also naturally handles cases where the desired end state evolves over time, such as when trying to hit an uncertain target where the target prediction evolves in real time.

The convex optimisation problem could be easily modified to achieve different goals, e.g. by removing the equality constraint on end velocity and acceleration, and adding the magnitude of the end velocity to the cost function, one solves a problem somewhat similar to that of [11].

Given some initial and final states, it would also be very useful to have an analytical method of calculating what the minimum required time horizon is. One possible solution would be to calculate the bang-bang trajectories as in [10].

The conservative nature of the jerk bounds means that only a fraction of the allowable body rates is typically used. Further work could be done to improve these bounds, e.g. using the unconstrained solution to find a better bound for the lowest thrust value produced along the trajectory. Alternatively, the optimisation problems could be solved in series and using the acceleration values from one solution to provide tighter (time varying) bounds for the next axis.

By combining all three axes into one optimisation problem that would now have 12 primary optimisation variables per stage, the constraints could be made even less conservative. The maximum thrust constraint (the right-hand side of (11)) can be encoded directly, noting that this is a convex quadratic constraint, which can also be solved efficiently. This would also allow rewriting the box constraint on jerk, (17), as a norm constraint, making it somewhat less conservative. Furthermore, by combining all axes in the optimisation problem, it becomes straight forward to define e.g. zones where the quadrocopter is allowed to fly (using the decoupled axes presented here, only zones aligned with the axes could be used). This has the major drawback of having to solve one large optimisation problem, instead of three small problems.

The problem can be made much less restrictive by allowing the end state to lie in a region "near" the desired end state, rather than equalling exactly the desired end state. The choice of the size of this region now becomes a design variable, and will depend on the application.

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`.
Specifically, we would like to thank Markus Hehn for the many discussions on optimal control and quadrocopter dynamics. We would also like to thank Alex Domahidi and Stephan Richter for their discussions on convex optimisation.

This research was supported by the Swiss National Science Foundation through grant agreement number 138112.

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] 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.

[4] K. Alexis, G. Nikolakopoulos, A. Tzes, and L. Dritsas, "Coordination of helicopter UAVs for aerial forest-fire surveillance," in *Applications of Intelligent Control to Engineering Systems*, 2009, vol. 39, pp. 169–193.

[5] N. E. Serrano, "Autonomous quadrotor unmanned aerial vehicle for culvert inspection," Ph.D. dissertation, Massachusetts institute of technology, 2011.

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

[7] P. Martin and E. Salaün, "The true role of accelerometer feedback in quadrotor control," in *IEEE International Conference on Robotics and Automation*, 2010, pp. 1623–1629.

[8] D. Mellinger, N. Michael, and V. Kumar, "Trajectory generation and control for precise aggressive maneuvers with quadrotors," in *International Symposium on Experimental Robotics*, 2010, pp. 664–674.

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

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

[11] M. Hehn and R. D. Andrea, "Real-time trajectory generation for interception maneuvers with quadrocopters," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, Villamoura, Portugal, 2012, pp. 4979–4984.

[12] F. Augugliaro, A. P. Schoellig, and R. D'Andrea, "Generation of collision-free trajectories for a quadrocopter fleet: a sequential convex programming approach," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, Villamoura, Portugal, 2012, pp. 1917–1922.

[13] P. Bouffard, A. Aswani, and C. Tomlin, "Learning-based model predictive control on a quadrotor: Onboard implementation and experimental results," in *IEEE International Conference on Robotics and Automation*, 2012, pp. 279–284.

[14] K. Alexis, C. Papachristos, G. Nikolakopoulos, and A. Tzes, "Model predictive quadrotor indoor position control," in *Mediterranean Conference on Control & Automation*, 2011, pp. 1247–1252.

[15] G. V. Raffo, M. G. Ortega, and F. R. Rubio, "MPC with nonlinear $\mathcal{H}_\infty$ control for path tracking of a quad-rotor helicopter," in *IFAC World Congress*, 2008, pp. 8564–8569.

[16] 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.

[17] S. Bouabdallah, P. Murrieri, and R. Siegwart, "Design and control of an indoor micro quadrotor," in *IEEE International Conference on Robotics and Automation*, 2004, pp. 4393–4398.

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

[19] M. Morari and J. H. Lee, "Model predictive control: past, present and future," *Computers and chemical engineering*, vol. 23, no. 4-5, pp. 667–682, 1999.

[20] J. Mattingley and S. Boyd, "CVXGen: a code generator for embedded convex optimization," *Optimization and Engineering*, vol. 13, no. 1, pp. 1–27, 2012.

[21] A. Domahidi, A. Zgraggen, M. Zeilinger, M. Morari, and C. Jones, "Efficient interior point methods for multistage horizons arising in receding horizon control," in *Proceedings of the 51st IEEE Conference on Decision and Control*, 2012, pp. 668–674.

[22] F. Ullman, "A Matlab toolbox for C-code generation for first order methods," Master's thesis, ETH Zurich, Zurich, Switzerland, 2011.

[23] J. Lofberg, "YALMIP : a toolbox for modeling and optimization in MATLAB," in *IEEE International Symposium on Computer Aided Control Systems Design*, September 2004, pp. 284 –289.

[24] M. Hehn, R. Ritz, and R. DAndrea, "Performance benchmarking of quadrotor systems using time-optimal control," *Autonomous Robots*, vol. 33, pp. 69–88, 2012.