International Conference on Unmanned Aircraft Systems (ICUAS), Miami, FL, USA, June 2017.

Fast Full State Trajectory Generation for Multirotors

Marius Beul and Sven Behnke

***Abstract*— Micro aerial vehicles, such as multirotors, are developed for applications like autonomous monitoring, inspection, and surveillance. Most of the current application scenarios assume a stationary environment and thus, trajectory generation relies on static targets. In this paper, we address time-optimal trajectory generation in dynamic environments, e.g., for landing on a moving platform.**

We extend our existing trajectory generation method by the ability to synchronize several axes of motion, following a master/slave approach for individual axes. We further explicitly treat moving targets, like moving landing platforms by planning in a target-centric frame. Our evaluation demonstrates the performance of the method under disturbances.

These results have application in dynamic multicopter flight, and also allow for fast and precise multicopter motion under challenging conditions.

I. Introduction

Micro aerial vehicles (MAVs) are becoming a key factor in reducing the required time, risks, and costs for, e.g., search and rescue missions, inspection tasks, and aerial photography. Due to their flexibility and low cost, they constitute a promising alternative to employing heavy machinery or even risking the health of humans in many situations.

Our research is performed in the context of the Mohamed Bin Zayed International Robotics Challenge[1] (MBZIRC) which poses several tasks that require fast and precice MAV flight under challenging conditions.

On one hand, MAV flight needs to be precise to pick up objects that are located on the ground, while rejecting disturbances posed by the environment like, e.g., wind gusts or by the picked objects themselves. On the other hand, MAV flight has to be robust against noisy sensor measurements, as no exact external measurement system is available. Since some target objects move with up to $5\,\frac{\mathrm{km}}{\mathrm{h}}$ in random directions, the MAV also has to react instantaneously to changing target velocities.

Another task is to land the MAV on a moving landing platform inside a circle with only $1\,\mathrm{m}$ diameter, while the platform moves with up to $15\,\frac{\mathrm{km}}{\mathrm{h}}$ which requires precisely approaching trajectory generation methods. Since the overall MBZIRC ranking depends on total task completion time, we aim for a fast motion of the MAV, but since the maximum allowed MAV velocity is $30\,\frac{\mathrm{km}}{\mathrm{h}}$, trajectory generation has to respect this constraint.

This work has been supported by a grant of the Mohamed Bin Zayed International Robotics Challenge (MBZIRC), the Bundesministerium für Wirtschaft und Energie in the Autonomics for Industry 4.0 project InventAIRy, and grant BE 2556/8-2 of the German Research Foundation (DFG).

The authors are with the Autonomous Intelligent Systems Group, Computer Science VI, University of Bonn, Germany mbeul@ais.uni-bonn.de

[1] http://www.mbzirc.com/

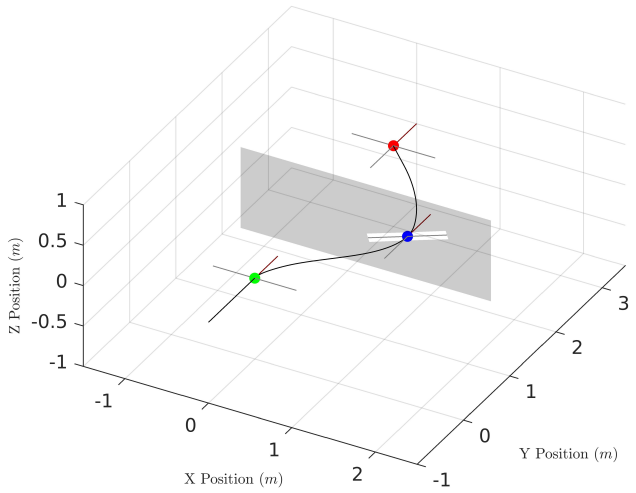


Fig. 1. Our MAV suddenly recieves an additional via state that it has to pass on its way to its target. A new trajectory is generated from the current state that incorporates state constraints (3D position, velocity and acceleration) in the via state as well as in the target waypoint. The trajectory respects limits on maximum allowed velocity and acceleration. For proper visualization, we magnified the pitch angle eight times. For simplicity, we use the following constraints for all figures: initial states are marked with a green dot, via states with a blue dot and target states with a red dot. Grey lines visualize the roll angle. The pitch angle is visualized by a grey and a red line.

In this paper, we extend our existing trajectory generation method [1] that generates time-optimal trajectories from the simplified dynamics of the MAV to address the specific problems posed in the MBZIRC. Due to the fast runtime, our method is not only able to compute offline trajectories, but is also capable of controlling position, velocity, and acceleration of the MAV in real-time. Since the method works nearly parameterless, no cumbersome parameter identification is needed and model uncertainties or changes in model parameters (e.g., when picking up objects) do not have to be modeled explicitly.

The key contributions of our paper are:

- improvement of our existing method to generate time-optimal trajectories with the ability to synchronize an arbitrary number of axes,
- extension of the method for state interception of moving targets by predicting the target velocity and acceleration,
- generation of trajectories that start from an inadmissible state and return the MAV into a permissible state, and
- reduction of the computational costs.

II. Related Work

Trajectory generation for MAVs is an active field in scientific research. The state of the art can be subdivided into

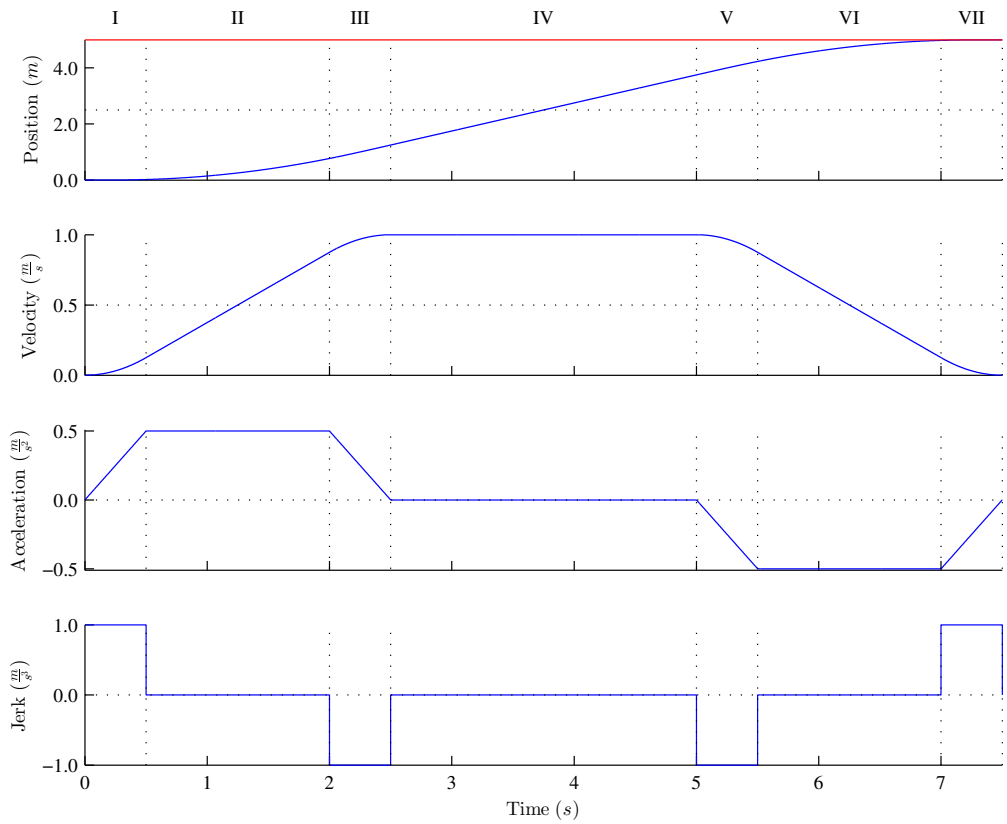


Fig. 2. Time-optimal trajectory generated with our method. Starting from state $\boldsymbol{x} = (0.0, 0.0, 0.0)$ (starting at the origin, zero velocity, zero pitch), it brings the simulated MAV to state $\boldsymbol{x}_{wayp} = (5.0, 0.0, 0.0)$ (5.0 m, zero velocity, zero pitch). The calculated switching times are $t_1 = 0.5\,\mathrm{s}$, $t_2 = 1.5\,\mathrm{s}$, $t_3 = 0.5\,\mathrm{s}$, $t_4 = 2.5\,\mathrm{s}$, $t_5 = 0.5\,\mathrm{s}$, $t_6 = 1.5\,\mathrm{s}$, and $t_7 = 0.5\,\mathrm{s}$.

approaches that either assume that it is possible to change the MAV velocity instantaneously (e.g., Nieuwenhuisen et al. [2]), or fully incorporate the dynamics of the MAV. As our approach belongs to the latter, we focus on dynamic trajectory generation approaches. Trajectories are either generated offline in a more long-term application, or in a short-term way for online usage in real-time feedback systems like Model Predictive Control (MPC).

The first category of methods includes the work of Motonaka et al. [3], who use kinodynamic planning. Due to the complex model parameter estimation and the computationally expensive nature of kinodynamic planning, the practical applicability of this approach is limited. Richter et al. [4] use iterative refinement of polynomials after finding a valid trajectory via straight-line rapidly exploring random trees (RRT). As the RRT needs $3\,\mathrm{s}$ to compute ($t < 1\,\mathrm{ms}$ for the iterative polynomial refinement), also their approach is not real-time capable.

A hybrid system, using offline and online trajectory generation was developed by Brescianini et al. [5]. They use the ACADO toolbox to generate pole-throwing maneuvers offline and execute them open-loop. Simultaneously, they generate a minimum snap real-time trajectory for catching the same pole. The catching maneuver does not consider state constraints, e.g., maximum velocity.

MPC-like trajectory generators have been developed by several groups. For example, Tomic et al. [6], Van Loock et al. [7], Ritz at al. [8], and Kahale et al. [9] use a first-principles model similar to our work. The trajectory generation problem is solved via numerical optimization by the first three works, and with a Matlab nonlinear program solver (direct collocation approach) by the latter work. Each of these works relies on the differential flatness of the MAV model. Also Chamseddine et al. [10] invenstigate the flatness property of the quadrotor model. The presented trajectory generation method considers actuator constraints but it is unclear if the proposed method is sufficiently fast to run in real time. Hehn et al. [11] replan time-optimal trajectories at every time step, making the method similar to an MPC. The approach assumes direct line of sight between current and final state. Furthermore, it is not able to incorporate velocity constraints, and is only able to generate trajectories targeting the vehicle at zero speed.

The works of Achtelik et al. [12] and Mellinger et al. [13] use N^{th} order polynomials with more degrees of freedom than equality constraints and a numerical solver to generate MPC-like trajectories. Boeuf et al. [14] employ splines in a decoupled global and local planner, resulting in a bang-singular-bang optimal trajectory, similar to our work. The solution, however, is not analytical.

In their works [15] and [16], Mueller et al. present a trajectory generator, capable of approaching the full target

TABLE I

SECOND-ORDER CONDITIONS FOR UNSYNCHRONIZED TRAJECTORIES.

Case 1	Case 2	Case 3	Case 4	Case 5	Case 6	Case 7	Case 8
$\theta_1 = \theta_{max}$	$\theta_1 = \theta_{max}$	$t_2 = 0$	$\theta_1 = \theta_{max}$	$t_2 = 0$	$t_2 = 0$	$\theta_1 = \theta_{max}$	$t_2 = 0$
$\theta_5 = \theta_{min}$	$\theta_5 = \theta_{min}$	$\theta_5 = \theta_{min}$	$t_6 = 0$	$t_6 = 0$	$\theta_5 = \theta_{min}$	$t_6 = 0$	$t_6 = 0$
$v_3 = v_{max}$	$t_4 = 0$	$v_3 = v_{max}$	$v_3 = v_{max}$	$v_3 = v_{max}$	$t_4 = 0$	$t_4 = 0$	$t_4 = 0$

state (position, velocity, and acceleration). Analogue to our approach, they use jerk (respectively the rotational velocity ω) as system input, but the convex optimization problem is solved numerically.

Kröger [17] presents a state-to-state solver for polynomial trajectories of robot manipulators with acceleration (Type II) or jerk (Type IV) as input. The algorithm runs with a typical cycle time of $t \leq 1\,\mathrm{ms}$, but no worst case or guaranteed cycle time is given. The Type IV solver, corresponding to the work presented here, is not publicly available. Haschke et al. [18] also plan minimum jerk trajectories. The authors assume the final velocity and acceleration to be zero, a restriction we do not make in this work. This renders target interception like we propose impossible. Also their solution takes $360\,\mu\mathrm{s}$ to compute and numerical instabilities occur under extreme input conditions.

Ezair et al. [19] compare polynomial trajectory generation algorithms regarding the order, state constraints, and constraints on initial and final conditions. No optimal trajectory generation method like ours (order 3 and general initial and final states) is mentioned.

Autonomous landing with MAVs was previously addressed by Bi et al. [20]. They use PID position control with visual feedback to land on a moving pushcart carrier.

To the knowledge of the authors, there exists no optimal analytical trajectory generation method for MAVs. Existing methods, like PID-trajectory tracking, lead to non-optimality and cannot handle nonlinearities like state constraints.

III. TRAJECTORY GENERATION

We state the trajectory generation problem as follows:
Find a trajectory from an arbitrary initial state to an arbitrary final state that satisfies state constraints such as maximum allowed velocity and acceleration and dynamic constraints such as smoothness of the position p, velocity v, and pitch θ and minimizes total time T. We assume that the environment around the MAV is obstacle free or that suitable via states are generated by a higher-level planer that prevents collisions with nearby obstacles like e.g. the method used in our previous work [21].

A. MAV Model

We assume the MAV to follow rigid-body dynamics and simplify it as a point mass with jerk j as system input. Following Newton's second law, the system is a triple integrator in each dimension with position p, velocity v, acceleration a, and jerk j,

$$\dot{p} = v, \qquad \dot{v} = a, \qquad \dot{a} = j. \tag{1}$$

TABLE II

SECOND-ORDER CONDITIONS FOR SYNCHRONIZED TRAJECTORIES.

Case 1	Case 3	Case 4	Case 5
$\theta_1 = \theta_{max}$	$t_2 = 0$	$\theta_1 = \theta_{max}$	$t_2 = 0$
$\theta_5 = \theta_{min}$	$\theta_5 = \theta_{min}$	$t_6 = 0$	$t_6 = 0$
$T = T_{traj.}$	$T = T_{traj.}$	$T = T_{traj.}$	$T = T_{traj.}$

Thus, the three-dimensional allocentric state of the MAV $\boldsymbol{x}$ can be expressed by

$$\boldsymbol{x} = \begin{pmatrix} p_x & p_y & p_z \\ v_x & v_y & v_z \\ a_x & a_y & a_z \end{pmatrix}. \tag{2}$$

We assume jerk j to be the direct control input to the system. In our previous work [1], we describe how the true control input pitchrate $\omega = \dot{\theta}$ is mapped to the input j. We further describe our linearization approach and total thrust computation. We also justify the applicability of our model. We do not consider the yaw axis since the dependence of the flat outputs of the quadrotor model and yaw is trivial.

B. Trajectory Composition

We decompose each one-dimensional trajectory into seven parts that either yield maximum pitch rate ($\omega = \omega_{max} \Leftrightarrow j = j_{max}$), zero pitch rate ($\omega = 0 \Leftrightarrow j = 0$), or minimum pitch rate ($\omega = \omega_{min} \Leftrightarrow j = j_{min}$). This is illustrated in Fig. 2 for a trajectory with start state $\boldsymbol{x} = \left(\begin{smallmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{smallmatrix}\right)$ and target state $\boldsymbol{x}_{wayp} = \left(\begin{smallmatrix} 5 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{smallmatrix}\right)$.

For every part n of the trajectory, we formulate a system of three differential equations, reflecting the state at the end of the part, including acceleration a, velocity v, and position p:

$$a_n = a_{n-1} + \int_0^{t_n} j_n \,\mathrm{d}t, \tag{3}$$

$$v_n = v_{n-1} + \int_0^{t_n} a_n \,\mathrm{d}t, \tag{4}$$

$$p_n = p_{n-1} + \int_0^{t_n} v_n \,\mathrm{d}t. \tag{5}$$

Since only 21 equations and 31 unknown variables are defined ($t_1, ..., t_7,\ a_0, ..., a_7,\ v_0, ..., v_7,\ p_0, ..., p_7$), we need second-order conditions to fully define the trajectory. We therefore assume $\boldsymbol{x}$ to be the current/start state a_0, v_0, p_0 and $\boldsymbol{x}_{wayp}$ the target state a_7, v_7, p_7.

As justified in our previous work, we also assume $a_3 = 0$ and one of the constraint-sets in Tab. I. We also assume that

TABLE III

COMPLEXITY ANALYSIS (NUMBER OF MATHEMATICAL OPERATIONS).

Case	t_1	t_2	t_3	t_4	t_5	t_6	t_7	$n_{eq.}$
Case 1	3	14	1	89	1	13	2	1
Case 2	6	484	2	0	2	484	4	2
Case 3	4	0	14	87	1	14	2	1
Case 4	6	28	2	866	107	0	129	2
Case 5	234	0	190	704	214	0	258	4
Case 6	20	0	0*	0	37944	84	8	4
Case 7	8	96	0*	0	16	0	37952	4
Case 8	28	0	0*	0	18138	0	88	4
Degenerated Case 1	2	13	1	84	1	14	2	1
Degenerated Case 3	6	0	29	386	2	28	4	2
Degenerated Case 4	4	26	2	182	8	0	32	2
Degenerated Case 5	6	0	28	182	8	0	32	2
Synchronized Case 1	6	365	2	319	2	365	4	2
Synchronized Case 3	8	0	68197	124	4	80	8	4
Synchronized Case 4	8	96	4	140	16	0	68204	4
Synchronized Case 5	-	-	-	-	-	-	-	-

*Since t_3 and t_5 are linearly dependent, we condensed both calculations into one equation.

the trajectory follows the pictured bang-singular-bang jerk strategy $j_1 = j_7 = j_{max}$, $j_3 = j_5 = j_{min}$ and $j_2 = j_4 = j_6 = 0$.

There exist eight different cases that either assume that the MAV reaches either or both acceleration and velocity limits. For one specific start and target configuration, only one case generates a valid trajectory. All permuted second-order conditions necessary for all possible trajectory templates are shown in Tab. I.

We further mirror the trajectory templates to expand the reachable set to all possible state space configurations. So, with $j_1 = j_7 = j_{min}$, $j_3 = j_5 = j_{max}$ and $j_2 = j_4 = j_6 = 0$, we obtain 16 trajectory templates in total.

We use the MathWorks® Matlab Symbolic Math Toolbox to find an analytical solution to the problem stated. For further details and mathematical derivation of our method, we refer to [1].

For simplicity, when not explicitly stated otherwise, we use the following constraints for all figures: $v_{max} = -v_{min} = 1\,\frac{\mathrm{m}}{\mathrm{s}}$, $\theta_{max} = -\theta_{min} = 0.051\,\mathrm{rad} \Leftrightarrow a_{max} = -a_{min} = 0.5\,\frac{\mathrm{m}}{\mathrm{s}^2}$, $\omega_{max} = -\omega_{min} = 0.202\,\frac{\mathrm{rad}}{\mathrm{s}} \Leftrightarrow j_{max} = -j_{min} = 1\,\frac{\mathrm{m}}{\mathrm{s}^3}$.

C. Axis Synchronization

Although our MAV model only has three dimensions (axes X, Y, and Z), we aim for an arbitrary number of axes n to finish their respective trajectory at the same time. We address this problem by first finding the time-optimal trajectory for each axis as described before. We then determine the maximum time and define it as the trajectory finishing time. $T_{traj.} = \max(T_n)$ with $T_n = t_{n,1} + ... + t_{n,7}$.

To slow down execution, we override the assumption to reach maximum velocity as fast as possible as mentioned in Sec. III-B with the second-order conditions stated in Tab. II. This results in the maximum reached velocity by the trajectory not to be the maximum velocity, but the one that makes the specific trajectory take exactly $T_n = T_{traj.}$.

Since we do not alter the acceleration and deceleration phases, they are still time-optimal. Thus, only the phase of constant velocity t_4 is stretched. With this method, we maximize the time of constant velocity of each axis respectively. This results in a straight trajectory when possible since during constant velocity phases in all axes, the trajectory is uncurved. This property could not be achieved by simply lowering the maximum acceleration and deceleration phases.

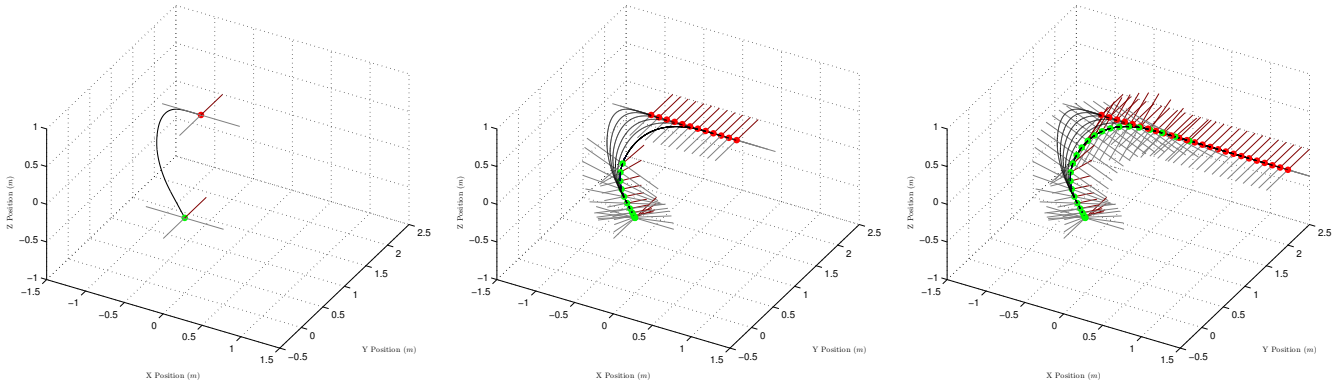
Since this method is only applicable for cases that reach the maximum velocity, we only override Case 1, Case 3, Case 4, and Case 5. One could also imagine to cancel the assumption of reaching the maximum pitch angle, but we refrain from doing so, since these trajectories only occur when start and target state are very close to another and thus unsynchronized behaviour is acceptable. Up to now, we found analytical solutions for Case 1, Case 3, and Case 4. We are currently working on finding an analytical solution for Case 5. When no synchronized trajectory can be found, we use the unsynchronized trajectory.

With synchronized trajectories, we are able to perform advanced flight maneuvers like depicted in Fig. 1. Since an arbitrary number of axes can be synchronized, one could also consider to synchronize the yaw axis or any number of additional axis (e.g. of a gimbal) if smooth motion is required.

D. Degenerated Trajectories

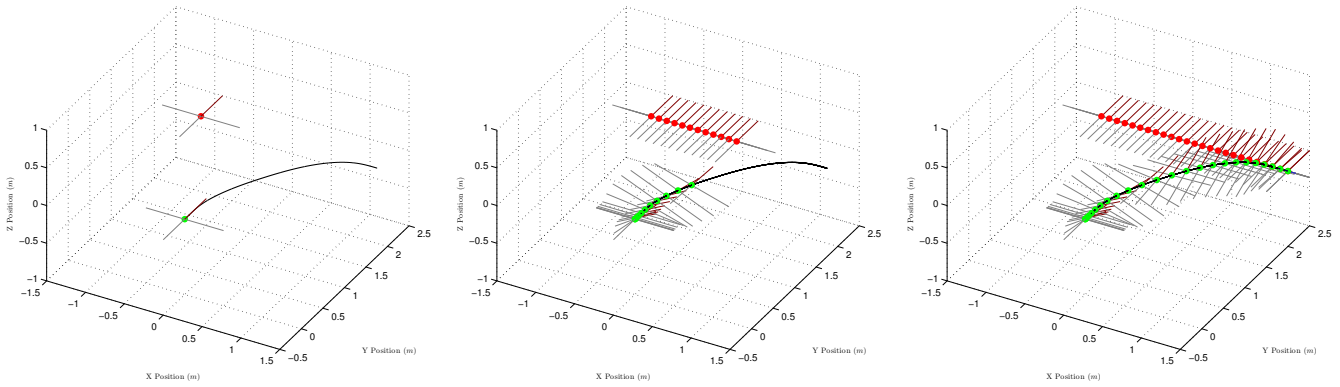
We extend out method proposed in [1], to generate degenerated trajectories that relax the assumption of a valid start state. Although under normal conditions, the MAV is always situated in a valid state space configuration, wind gusts or other disturbances can cause the MAV to leave the valid envelope.

Although pitch angles exceeding the maximum are very uncommon, they can be dangerous since the acceleration of the MAV is not bounded anymore. In this case, we simply stop our trajectory generation method and command the MAV to reach zero pitch. When the acceleration constraint



(a) (b) (c)

Fig. 3. Evolution of generated MAV trajectories when starting from $\boldsymbol{x} = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}$ with target $\boldsymbol{x}_{wayp} = \begin{pmatrix} -1+0.5 \cdot t & 2 & 0 \\ 0.5 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}$ after $0.2\,\mathrm{s}$ (a), $2.5\,\mathrm{s}$ (b), and $5.0\,\mathrm{s}$ (c) without prediction of the waypoint motion. New trajectories are planned every $200\,\mathrm{ms}$.



(a) (b) (c)

Fig. 4. Evolution of generated MAV trajectories when starting from $\boldsymbol{x} = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}$ with target $\boldsymbol{x}_{wayp} = \begin{pmatrix} -1+0.5 \cdot t & 2 & 0 \\ 0.5 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}$ after $0.2\,\mathrm{s}$ (a), $2.5\,\mathrm{s}$ (b), and $5.0\,\mathrm{s}$ (c) with prediction of the waypoint motion. New trajectories are planned every $200\,\mathrm{ms}$.

is not violated anymore, we continue planning trajectories with our method.

The second constraint that can be violated is the velocity constraint. When the velocity constraint is violated, the first priority is to bring the MAV back into the allowed state envelope. We address this issue by using a different jerk input strategy. For degenerated trajectories, we use the same second order conditions as stated in Tab. I, but instead of using a jerk input of $j_1 = j_7 = j_{max}$, $j_3 = j_5 = j_{min}$ and $j_2 = j_4 = j_6 = 0$, we use $j_1 = j_7 = j_{min}$, $j_3 = j_5 = j_{min}$ and $j_2 = j_4 = j_6 = 0$ as input. This means that, e.g., when starting with a too high velocity, we use a negative pitch to slow the MAV down to the maximum allowed velocity and stay at this velocity as long as possible. Finally, we use negative pitch again to bring the MAV to rest at the target.

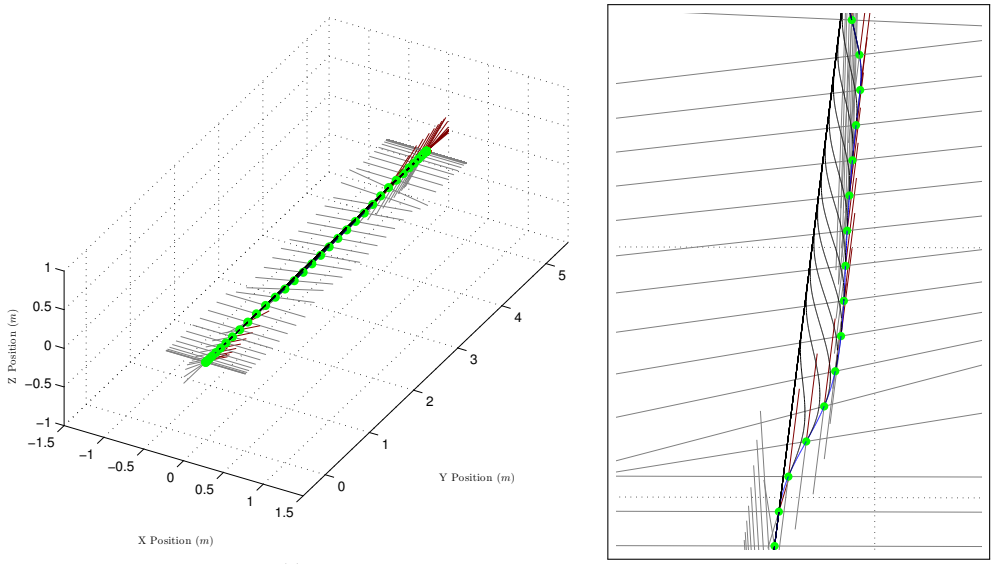
Since degenerated trajectories are only necessary for trajectories that reach v_{max}, only Case 1, Case 3, Case 4, and Case 5 are evaluated with the changed jerk strategy.

E. Interception

Our method is able to generate trajectories that connect arbitrary start and target states, satisfying dynamic constraints in an allocentric world-frame. Due to this property, we are able to plan feasible trajectories with arbitrary number of via states, e.g., shown in Fig. 1. Here, our method only specifies target position, velocity and acceleration, but makes no assumption about the motion of the waypoint itself.

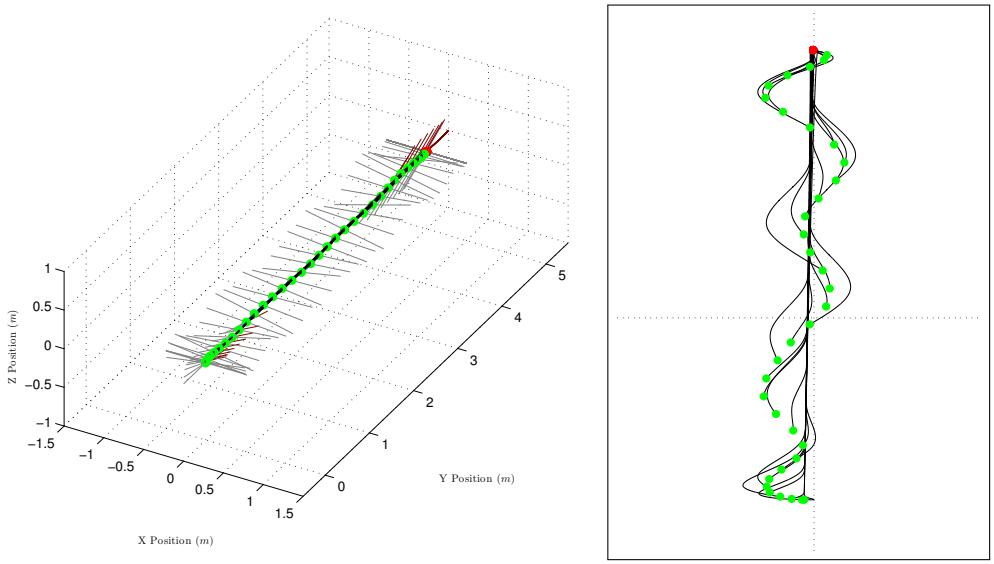
Another trajectory with nonzero target velocites is shown in Fig. 3a. It can be seen that the MAV lunges to exactly reach the desired velocity of $0.5\,\frac{\mathrm{m}}{\mathrm{s}}$ in X-direction. Nevertheless, since we do not explicitly consider the waypoint motion, the waypoint moves between replanning steps, resulting in a slightly different trajectory. Due to replanning, the MAV converges to the target without steady-state error. The MAV trajectory is suboptimal, however.

We address this problem by first predicting the target velocity and acceleration in an allocentric frame (here $v_{x,predict} = 0.5\,\frac{\mathrm{m}}{\mathrm{s}}, a_{x,predict} = 0\,\frac{\mathrm{m}}{\mathrm{s}^2}$). Afterwards, we transform all states including via states and all constraints into a moving target-centric coordinate system. Since the



(a) (b)

Fig. 5. During a 5 m flight, the MAV is disturbed by a lateral wind gust between 1.5 m and 4 m which results in an acceleration of $0.2\,\frac{\mathrm{m}}{\mathrm{s}^2}$. Immediately after the disturbance is measured, new trajectories are planned to compensate for the resulting unwanted lateral motion. New trajectories are planned every $t_{replan} = 200\,\mathrm{ms}$. Axis synchronization was switched off for this experiment. The whole trajectory can be seen in (a). A magnification of the specific replanned trajectories is shown in (b).



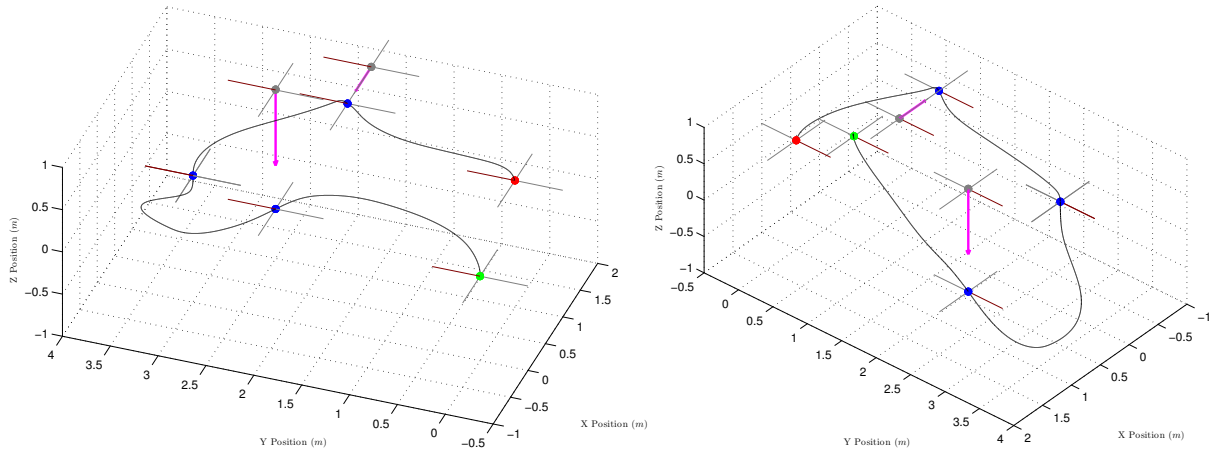
(a) (b)

Fig. 6. During a 5 m flight, lateral velocity estimation is noisy (zero mean, standard deviation $0.03\,\frac{\mathrm{m}}{\mathrm{s}}$). New trajectories are planned every 200 ms. Axis synchronization was switched off for this experiment. The whole trajectory can be seen in (a). A magnification of the specific replanned trajectories is shown in (b). For visualization purposes, we do not show the pitch and roll angle.

trajectory generation problem is symmetrical, the trajectory is generated as if the MAV would have an initial velocity of $v_x = -0.5\,\frac{\mathrm{m}}{\mathrm{s}}$ and initial acceleration $a_x = 0\,\frac{\mathrm{m}}{\mathrm{s}^2}$. Since constraints are defined symmetrical ($V_{min} = -V_{max}$) in the allocentric frame, the transformation into a moving frame also renders the constraints to become asymmetrical $V_{min} \neq -V_{max}$.

Fig. 3 and Fig. 4 show an experiment with target prediction inactive and active. It can be seen that with our method, an optimal intersection state is calculated so that successively replanned trajectories are lying on each other. By planning in the target-centric frame, the constant optimal interception state is implicitly projected in the allocentric world-frame. By employing this method, we can analytically generate optimal target interception trajectories without the use of numerical methods like, e.g., gradient descent.

We want to emphasize here that the interception is different than just reaching a non-stationary final state. The inter-



(a) (b)

Fig. 7. Our method is used to generate an aerobatic flight trajectory. Starting at the origin (green), the MAV flies 1) through a downward accelerating waypoint (grey). The waypoint starts at position $p_x = 1.0\,\mathrm{m}$, $p_y = 2.5\,\mathrm{m}$, $p_z = 1.0\,\mathrm{m}$ and accelerates in negative Z-direction with $-0.2\,\frac{\mathrm{m}}{\mathrm{s}^2}$ (long magenta arrow). The waypoint is intercepted with the maximum allowed velocity $v_y = 1.0\,\frac{\mathrm{m}}{\mathrm{s}}$. The actual passing position is $p_x = 1.0\,\mathrm{m}$, $p_y = 2.5\,\mathrm{m}$, $p_z = -0.41\,\mathrm{m}$ with velocity $v_z = -0.79\,\frac{\mathrm{m}}{\mathrm{s}}$. 2) The MAV flies through a stationary waypoint at position $p_x = 0.0\,\mathrm{m}$, $p_y = 3.0\,\mathrm{m}$, $p_z = 0.5\,\mathrm{m}$ with a velocity of $v_z = 0.3\,\frac{\mathrm{m}}{\mathrm{s}}$. 3) When passing the second waypoint, the third waypoint (grey) starts to move from position $p_x = 1.0\,\mathrm{m}$, $p_y = 1.5\,\mathrm{m}$, $p_z = 1.5\,\mathrm{m}$ in negative X-direction with constant velocity $-0.2\,\frac{\mathrm{m}}{\mathrm{s}}$ (short magenta arrow). The waypoint is intercepted with velocity $v_y = -0.3\,\frac{\mathrm{m}}{\mathrm{s}}$. The actual passing position is $p_x = 0.32\,\mathrm{m}$, $p_y = 1.5\,\mathrm{m}$, $p_z = 1.5\,\mathrm{m}$. 4) The MAV comes to rest at position $p_x = 1.0\,\mathrm{m}$, $p_y = 0.0\,\mathrm{m}$, $p_z = 0.5\,\mathrm{m}$ (red).

ception implicitly considers not only the kinematic properties of the trajectory, but also the movement of the target.

Please note that the predicted velocity and acceleration does not have to inevitably match the target velocity. Scenarios where the moving target has to be intercepted orthogonally, e.g., flying through a falling ring are also realisable with our method (see Fig. 7).

Due to the ability to replan very fast, the target state (3D position, velocity and acceleration) only has to be approximately known. Since trajectories are computed online, target state estimation errors are canceled with the next iteration of the planner. With perfect state estimation however, the planner can be run open-loop as shown in Fig. 7.

F. Computation Time

Although the computational costs of our method are marginal, we were able to reduce them further by reusing already calculated results and by initializing the replanning. In [1], all switching times were calculated independently, only depending on the start and target state. We now calculate the switching times in an order to maximize the use of already calculated switching times. By doing so, we can reduce the complexity of some equations. The number of mathematical operations needed to solve for a specific switching time are listed in Tab. III.

For every equation, there exist a number of solutions equal to the order of the equation. We were able to eliminate possible solutions by posing assumptions on some variables like $t_1, ..., t_7 > 0$, etc. Thus, we were able to reduce the number of equations to be solved for some of the considered cases. The number of candidate equations is also shown in Tab. III. Especially Case 6 benefits from our method since now only $38\,056$ instead of $1\,496\,111$ mathematical operations have to be conducted.

Since replanning typically happens with a high rate, start and target state often differ only marginally from the previous trajectory. Thus, the propabilty of the same case to be valid when replanning is high. We exploit this characteristic by determining the order in which to evaluate the cases by the previous trajectory. By doing so, we often do not need to solve all cases.

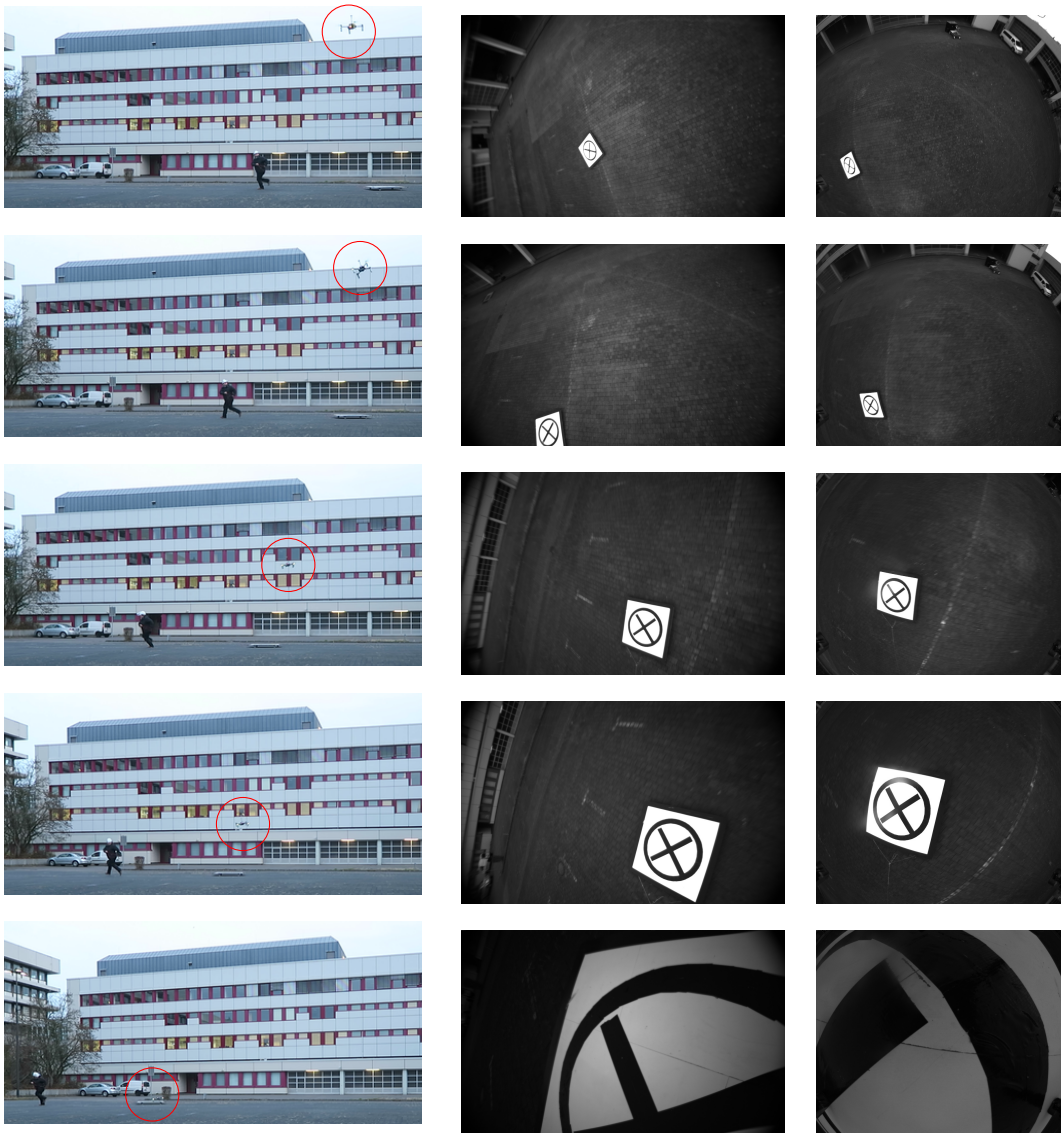
IV. Results

A. Disturbance Rejection

In order to evaluate the disturbance rejection capabilities, we disturb the MAV with a lateral wind gust that results in a lateral acceleration of $0.2\,\frac{\mathrm{m}}{\mathrm{s}^2}$. The experiment can be seen in Fig. 5. Immediately after the disturbance is measured, new trajectories are planned that bring the MAV back to the optimal path. Since we do not explicitly model disturbances, the generated trajectories assume a disturbance-free environment. This leads the MAV to settle in the equilibrium between planned trajectory and windgust and results in a steady-state error during the wind gust of approximately $2.3\,\mathrm{cm}$. We want to mention here that the error strongly depends on the replanning time. Although never fully eliminable, it can become very small with $t_{replan} \to 0$.

B. Measurement Noise Rejection

In order to assess the impact of measurement noise on the performance of our method, we added normally distributed noise with standard deviation of $0.03\,\frac{\mathrm{m}}{\mathrm{s}}$ on the velocity measurements in the lateral direction. The resulting trajectories are shown in Fig. 6. It can be seen that the MAV tries to compensate for the wrongly percieved velocity and thus leaves the optimal path. Since the measurement noise is zero mean, the impact on the system performance is only



(a) (b) (c)

Fig. 8. Landing on a moving target. (a) External view. (b) Image from the onboard camera. The camera is mounted in a 30° angle facing forward/downward. (c) Image from the downward facing fisheye camera. First, the MAV (circled red) is searching for the target. After the first detection, the MAV predicts the target motion and accelerates in direction of the optimal interception point. Axes are synchronized, so that the MAV decents in a straight line. The target velocity is approximately $3.5\,\frac{\mathrm{m}}{\mathrm{s}}$. The total landing procedure takes $4.3\,\mathrm{s}$.

temporary. The maximum accumulated position error in this experiment is $1.5\,\mathrm{cm}$.

C. Aerobatic Flight

In Fig. 7, we illustrate the performance of our method with multiple stationary and moving waypoints. When intercepting moving targets, the MAV adapts its velocity and acceleration, in direction of the movement, to the target. Thus, when passing through the first waypoint, the MAVs does not pass purely horizontal, but with the same downwards velocity and acceleration as the waypoint. By doing so, the trajectory does not rely on a fast motion through the waypoint, but can pass with arbitrary orthogonal velocities. Here, the orthogonal velocities are chosen so that the MAV could pass through, e.g., a moving ring. Videos of this simulation and other experiments can be found on our website[2].

D. Landing on a Moving Target

In order to validate our method, we conducted real robot experiments. Fig. 8 shows our DJI Matrice 100 MAV landing on a moving platform. The platform was pulled with an almost constant velocity of $3.5\,\frac{\mathrm{m}}{\mathrm{s}}$. After visually detecting the platform from a height of $6\,\mathrm{m}$, the proposed method is used to calculate a trajectory that guides the MAV to the optimal interception point. Subsequently, the trajectory is executed. Due to axis synchronization, the MAV flies on a straight glide path. The trajectory is replanned with $50\,\mathrm{Hz}$.

[2] http://www.ais.uni-bonn.de/videos/ICUAS_2017_Beul

Attitude commands are also sent to the MAV with $50\,\mathrm{Hz}$. Position and velocity estimates of the target are updated with $40\,\mathrm{Hz}$. The entire landing process is completed after only $4.3\,\mathrm{s}$

V. Conclusions

We proposed an analytical time-optimal trajectory generation method for MAVs that is able to run in real-time and thus can be used as MPC. Starting from a simple parameterless first-principles model of the MAV, we compute optimal switching times for the rotational velocity of the MAV, respecting state, and input constraints.

With the ability to specify the full state of the MAV, it is possible to target moving or even accelerating waypoints like, e.g., a moving landing platform.

We evaluated the effectiveness of the proposed approach, benchmarking it under disturbances and under state uncertainty. We further evaluated the computational costs of the method and show that it is computationally inexpensive. We showed that in general, our approach is capable of dealing with arbitrary input and state constraints. The method is able to recover the MAV from inadmissable start states and brings it back into the allowed state envelope.

Real-robot experiments were conducted to demonstrate the applicability of the proposed method in a real-world scenario. In future work, we want to conduct further real-robot experiments to prove the applicabilty of our method with different robots and to be able to give quantitative results of the performance of the proposed method. In particular, we want to utilize our method at the MBZIRC to benchmark it against methods used by other groups. Our method can be employed as is, or build upon.

References

[1] M. Beul and S. Behnke, "Analytical time-optimal trajectory generation and control for multirotors," in *Proc. of the Int. Conf. on Unmanned Aircraft Systems (ICUAS)*, 2016.

[2] M. Nieuwenhuisen and S. Behnke, "Layered mission and path planning for MAV navigation with partial environment knowledge," in *Proc. of the Int. Conf. on Intelligent Autonomous Systems (IAS)*, 2014.

[3] K. Motonaka, K. Watanabe, and S. Maeyama, "Kinodynamic motion planning and control for an x4-flyer using anisotropic damping forces," in *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 2013.

[4] C. Richter, A. Bry, and N. Roy, "Polynomial trajectory planning for agressive quadrotor flight in dense indoor environments," in *Proc. of the Int. Symposium of Robotics Research (ISRR)*, 2013.

[5] D. Brescianini, M. Hehn, and R. D'Andrea, "Quadrocopter pole acrobatics," in *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 2013.

[6] T. Tomić, M. Maier, and S. Haddadin, "A computationally efficient algorithm for state-to-state quadrocopter trajectory generation and feasability verification," in *Proc. of the IEEE Int. Conf. on Robotics and Automation (ICRA)*, 2014.

[7] W. V. Loock, G. Pipeleers, and J. Swevers, "Time-optimal quadrotor flight," in *Proc. of the European Control Conference (ECC)*, 2013.

[8] R. Ritz, M. Hehn, S. Lupashin, and R. D'Andrea, "Quadrocopter performance benchmarking using optimal control," in *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 2011.

[9] E. Kahale, P. C. Garcia, and Y. Bestaoui, "Minimum time reference trajectory generation for an autonomous quadrotor," in *Proc. of the Int. Conf. on Unmanned Aircraft Systems (ICUAS)*, 2014.

[10] A. Chamseddine, Y. Zhang, C. A. Rabbath, C. Join, and D. Theilliol, "Flatness-based trajectory planning/replanning for a quadrotor unmanned aerial vehicle," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 48, no. 4, pp. 2832–2848, 2012.

[11] M. Hehn and R. D'Andrea, "Quadrocopter trajectory generation and control," in *IFAC World Congress*, 2011.

[12] M. W. Achtelik, S. Lynen, M. Chli, and R. Siegwart, "Inversion based direct position control and trajectory following for micro aerial vehicles," in *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 2013.

[13] D. Mellinger and V. Kumar, "Minimum snap trajectory generation and control for quadrotors," in *Proc. of the IEEE Int. Conf. on Robotics and Automation (ICRA)*, 2011.

[14] A. Boeuf, J. Cortés, R. Alami, and T. Siméon, "Planning agile motions for quadrotors in constrained environments," in *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 2013.

[15] M. W. Mueller and R. D'Andrea, "A model predictive controller for quadrocopter state interception," in *Proc. of the European Control Conference (ECC)*, 2011.

[16] M. W. Mueller, M. Hehn, and R. D'Andrea, "A computationally efficient algorithm for state-to-state quadrocopter trajectory generation and feasability verification," in *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 2013.

[17] T. Kröger, "Opening the door to new sensor-based robot applications - the reflexxes motion libraries," in *Proc. of the IEEE Int. Conf. on Robotics and Automation (ICRA)*, 2011.

[18] R. Haschke, E. Weitnauer, and H. Ritter, "On-line planning of time-optimal, jerk-limited trajectories," in *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 2008.

[19] B. Ezair, T. Tassa, and Z. Shiller, "Planning high order trajectories with general initial and final conditions and asymmetric bounds," *The Int. Journal of Robotics Research*, vol. 33, no. 6, pp. 898–916, 2014.

[20] Y. Bi and H. Duan, "Implementation of autonomous visual tracking and landing for a low-cost quadrotor," *Optik - International Journal for Light and Electron Optics*, vol. 124, no. 18, 2013.

[21] M. Nieuwenhuisen, D. Droeschel, M. Beul, and S. Behnke, "Autonomous navigation for micro aerial vehicles in complex gnss-denied environments," vol. 84, no. 1, pp. 199–216, 2016.