IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Daejeon, Korea, October 2016.

Local Multiresolution Trajectory Optimization for Micro Aerial Vehicles Employing Continuous Curvature Transitions

Matthias Nieuwenhuisen and Sven Behnke

***Abstract*— Complex indoor and outdoor missions for autonomous micro aerial vehicles (MAV) require fast generation of collision-free paths in 3D space. Often not all obstacles in an environment are known prior to the mission execution. Consequently, the ability for replanning during a flight is key for success. Our approach locally optimizes trajectories of grid-based path planning. It preserves obstacle-freeness of the path and ensures smoothness with continuous curvature transition segments.**

Fast optimization and frequent reoptimization is made possible by means of local multiresolution time discretization. With our extensions, high dimensional flight trajectories incorporating velocities and accelerations can be planned with a time discretization of 100 Hz within the prediction horizon of the underlying controller.

I. Introduction

The application of micro aerial vehicles (MAVs) is nowadays considered for an increasing number of tasks. Still, for most MAVs employed in real world applications a human pilot is necessary to teleoperate the vehicle to ensure safe operation in the vicinity of obstacles. Another option is to operate in complete open space, e.g., at sufficient height, and let the MAV fly to waypoints without considering obstacles. It is desirable to let MAVs follow more elaborated trajectories to reach goals and avoid obstacles without the requirement to pass waypoints manually defined during mission specification.

The real-time generation of highly dynamic trajectories is mostly performed in free space inside of motion capture volumes, e.g., to quickly reach also dynamically changing goal states [1]. Other approaches plan collision-free trajectories in advance and execute those with high precision [2]. These approaches assume complete knowledge of the environment and very reliable control of the MAV. In most application scenarios outside of a controlled lab, the environment can change unpredictably or acquiring a model of the environment itself is the mission objective. Thus, closing the gap between conservative flying in free space and dynamic trajectory following is key to expand the application domain of MAVs.

Continuous perception of the environment and tracking the MAV state is a prerequisite for safe operation. To react on the perceived changes timely, frequent adaption of the planned flight trajectory is inevitable. When aiming at fast trajectory execution, the flight dynamics have to be taken into account in addition to spatial constraints. To reduce the complexity of this kinodynamic planning problem, we follow a multistage approach from a 3D spatial planner to dynamic 12D trajectory optimization.

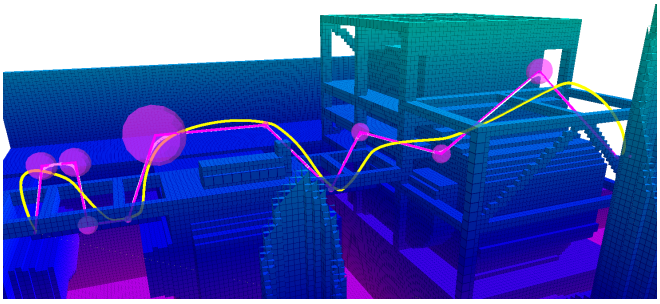

Fig. 1. Processing steps of trajectory optimization. To compute dynamically feasible trajectories, we initialize the optimization with a collision free path of straight line segments (violet). We construct continuous curvature transition segments (white) bound by obstacle-free spheres (violet spheres). After estimating velocities and accelerations along the trajectory with a simple MAV dynamics model, we optimize this initial trajectory to a dynamically feasible local optimum preserving the topological constraints (yellow).

To obtain smooth collision-free trajectories, we use the gradient-based trajectory optimizer CHOMP [3]. Trajectory optimization is prone to getting stuck in local minima. Hence, a good initialization is necessary. For initialization, we plan coarse feasible 3D paths using a grid-based path planner. Although, these coarse plans prevent an optimizer to get stuck in local minima that yield unfeasible trajectories, they are far from smooth and lack velocity and acceleration dimensions. This leads to longer optimization times—too long for frequent replanning.

We mitigate the influence of suboptimal initialization by modifying the planned path locally. By determining convex volumes of free space around waypoints, we can safely replace parts of the planned path with dynamically more feasible transition segments inside these volumes. Furthermore, we approximate optimal velocities and accelerations with a simple MAV dynamics model before optimization. Fig. 1 shows an example solution of our optimization. To make frequent replanning feasible, we reoptimize trajectories and use a local multiresolution discretization of time along the trajectory. As result it is computationally feasible to plan at the control rate for the prediction horizon of a model predictive controller and at lower resolution to the global goal configuration.

This work is partially funded by the German Research Foundation (DFG) in the project Mapping on Demand (grant agreement BE 2556/8-2) and by the European Commission in the FP7 project EuRoC (grant agreement 608849).

The authors are with the Autonomous Intelligent Systems Group, Institute for Computer Science VI, University of Bonn, Germany nieuwenh at ais.uni-bonn.de

II. Related Work

Heng et al. [4] use a multiresolution grid map to represent the surroundings of a quadrotor. A feasible plan is generated with a vector field histogram. Schmid et al. [5] autonomously navigate to user specified waypoints in a mine. The map used for planning is created by an onboard stereo camera system. By using rapidly exploring random belief trees (RRBT), Achtelik et al. [6] plan paths that do not only avoid obstacles, but also minimize the variability of the state estimation. Recent search-based methods for obstacle-free navigation include work of MacAllister et al. [7]. They use A* search to find a feasible path in a four-dimensional grid map. They also incorporate the asymmetric shape of their MAV. Cover et al. [8] use a search-based method as well.

To plan high-dimensional trajectories, often sampling-based planners are employed, including KPIECE [9] and randomized kinodynamic planning [10]. In addition to those sampling-based motion planning algorithms, trajectory optimization allows for efficient generation of high-dimensional trajectories. Covariant Hamiltonian Optimization and Motion Planning (CHOMP) is a gradient-based optimization algorithm proposed by Ratliff et al. [3]. It uses trajectory samples, which initially can include collisions, and performs a covariant gradient descent by means of a differentiable cost function to find an already smooth and collision-free trajectory. A planning algorithm based on CHOMP is the Stochastic Trajectory Optimization for Motion Planning (STOMP) by Kalakrishnan et al. [11]. In contrast to CHOMP, it is no longer required to use cost functions for which gradients are available, while the performance stays comparable. Another algorithm derived from CHOMP is ITOMP, an incremental trajectory optimization algorithm for real-time replanning in dynamic environments [12]. In order to consider dynamic obstacles, conservative bounds around them are computed by predicting their velocity and future position. Since fixed timings for the trajectory waypoints are employed and replanning is done within a time budget, generated trajectories may not always be collision-free.

Augugliaro et al. [13] compute collision-free trajectories for multiple MAVs simultaneously. Other obstacles than the MAVs are not considered here. Similar to our approach, Richter et al. [2] plan MAV trajectories in a low dimensional space (using RRT*) and optimize the trajectory with a dynamics model afterwards to achieve short planning times. Our approach does not have the constraint that the optimized path has to include the planned waypoints. Another approach using optimization by means of polynomial splines between waypoints focuses on time-optimal trajectories computed in real-time (Bipin et al. [14]). Collisions are avoided by intermediate waypoints from a high-level planner and are not explicitly considered in the optimization process. Andreasson et al. [15] employ optimization to compute steerable trajectories for automated ground vehicles.

Majumdar and Tedrake use compositions of preprocessed trajectories to generate flight paths that are safe under uncertainty in real-time [16]. In contrast, we frequently modify a trajectory in real-time to react on changes in the environment and uncertain path execution.

In terms of motion planning for end-effectors, He et al. perform global multiresolution optimization with multigrid CHOMP [17]. In contrast to our approach, the whole trajectory is first optimized at low resolution and subsequently refined by adding intermediate points, yielding a globally optimized trajectory in high resolution. They use the property that the optimization converges faster, if an already near-optimal trajectory is given as initialization. In contrast, we optimize in high resolution only in the close proximity to the robot and refine the future trajectory as the robot unrolls it.

III. Problem Formulation

The static state of an MAV is a 6-tuple of a 3D position $p = (x, y, z)$ and a 3D rotation $r = (roll, pitch, yaw)$. Although in general poses of the MAV are six dimensional, for multirotors only four dimensions can be controlled independently. The roll and pitch angles directly influence the horizontal acceleration of multirotors. Thus, our start and goal poses are 4D tuples (x, y, z, θ) with a 3D position and yaw-rotation θ. Here, we assume that velocity and acceleration at start and goal pose are zero, even though this assumption can be relaxed.

To control our MAV, we employ a linear model predictive controller (MPC) with a prediction horizon of 20 time steps. The controller operates at 100 Hz resulting in a prediction horizon of 200 ms. Input to the controller is an 8D trajectory containing 4D positions and the corresponding velocities.

To generate dynamically feasible trajectories, we formulate trajectory planning as an optimization problem. Accordingly, the goal is to find a trajectory, which minimizes the costs calculated by a predefined cost function. As an input, the trajectory optimizer gets a start and a goal configuration $x_0 = (p_0^x, p_0^y, p_0^z, \theta_0)^\top, x_N = (p_N^x, p_N^y, p_N^z, \theta_N)^\top \in \mathbb{R}^4$. The output of the algorithm is a trajectory $\Theta \in \mathbb{R}^{4\times N+1}$ consisting of one trajectory vector $\Theta^d = (x_0^d, \ldots, x_N^d)^\top \in \mathbb{R}^{N+1}$ per dimension d, discretized into $N + 1$ waypoints. To be directly executable by the MAV controller, in general the trajectory points need to have a fixed duration Δt = 10 ms. We relax this assumption in Sec. V. Besides a cost function, the trajectory optimizer has to be initialized with an initial trajectory, e.g., an interpolation between start and goal configuration. The optimization problem we solve iteratively is defined by

$$\min_{\Theta} \left[\sum_{i=0}^{N} q(\Theta_i) + \sum_{d} \frac{1}{2} {\Theta^d}^\top \mathbf{R} \Theta^d \right]. \tag{1}$$

Here, $q(\Theta_i)$ is a predefined cost function calculating the costs for each state in Θ, ${\Theta^d}^\top \mathbf{R} \Theta^d$ is the sum of control costs along the trajectory in dimension d with $\mathbf{R}$ being a matrix representing control costs. The trajectory optimizer now attempts to solve the defined optimization problem by means of the gradient-based optimization method CHOMP [3].

The cost function $q(\Theta_i)$ is a weighted sum of I) piecewise linear increasing costs c_o induced by the proximity to obstacles, II) squared costs c_a caused by acceleration limits, and III) squared costs c_v caused by velocity constraints. The obstacle costs c_o increase linear with a slope o_{far} from a maximum safety distance to a minimum safety distance plus a margin. From the minimum safety distance plus a margin to the obstacle, the costs increase with a steeper slope o_{close} to allow for gradient computation in the vicinity of obstacles.

Velocities and accelerations as derivatives of the state are implicitly modeled by the duration between discretization steps. The trajectory optimization converges faster when the initialization is close to the (locally) optimal trajectory. This includes velocities and accelerations. Even though the optimal solution is naturally not known in advance, we can make some assumptions about the MAV dynamics that reduce the convergence time and avoid unfeasible local minima.

IV. Initialization

Initialization of the optimizer with a control cost-optimal interpolation between start and goal configuration leads to trajectories that converge to a non collision-free local optimum in general. Thus, to find valid paths between start and goal configuration, we plan collision-free paths employing an A* path planner in a low-dimensional subspace of only the 3D translational part of the trajectory. The velocities and accelerations along this trajectory are estimated employing a simple motion model of our MAV [18]. As our MAV is approximately rotational symmetric, we omit planning for the heading and initialize the yaw trajectory with a control cost-optimal interpolation.

Transition Between Path Segments The translational part of the planned flight paths is a sequence of straight line segments connecting start and goal position. By employing a grid-based planner, the transition between consecutive segments is restricted to a few discrete angles. After simplification of the initial plan, the discretization effects are mitigated, but still present, resulting in discontinuities in its derivatives—most importantly velocity and acceleration. These discontinuities cause gradients of large magnitude during optimization. Thus, a lot of optimization effort is spent on smoothing these transitions instead of optimizing the continuous parts of the trajectory. We address this by introducing connections with continuous curvature between straight segments.

To assure that the modified path remains collision-free without costly exact planning of the connecting segments, we calculate spheres containing solely free space around the segment transitions. The radius of a sphere around the transition between path segment s_{i-1} to s_i is calculated as $r_i = min(\{\frac{1}{2}\|s_{i-1}\|, \frac{1}{2}\|s_i\|, d_o(i)\})$, where $d_o(i)$ is the distance to the nearest obstacle. The start and end points of the connecting segment are the intersection points of the planned path with the sphere. We construct the connecting segment in a planar subspace defined by start point, end point, and transition point. In the following, we will use 2D coordinates on this plane.

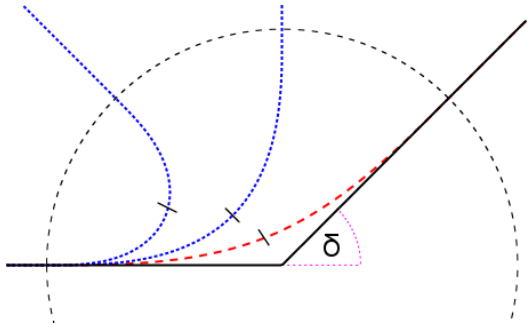

Fig. 2. Construction of transition segments. Transition segments (red dashed line) connect two straight line path segments (black solid lines) with continuous curvature inside a bounding sphere restricted by the distance to obstacles and other waypoints. Their shape is defined by heading change δ (violet dotted lines), their scale by the radius of the bounding sphere. The segments are constructed by two clothoids mirrored in the center. Transitions for sharper heading changes of are depicted by blue dotted lines.

As connecting segments, we employ clothoids—also often referred to as Euler spirals. Clothoids are curves with linear increasing curvature $\kappa(l) = l\sigma$ along the curve. The Cartesian coordinates are given by

$$x(l) = \sqrt{\frac{\pi}{\sigma}} C\left(\sqrt{\frac{l^2\sigma}{\pi}}\right), y(l) = \sqrt{\frac{\pi}{\sigma}} S\left(\sqrt{\frac{l^2\sigma}{\pi}}\right),$$

with $C(x) = \int_0^x \cos(\frac{\pi}{2}t^2)dt$ and $S(x) = \int_0^x \sin(\frac{\pi}{2}t^2)dt$ and sharpness σ.

To calculate the sharpness parameter for our clothoid segment, we follow ideas from [19]. The parameter σ for a normalized clothoid depends on the change in heading between start and end of the clothoid segment. Due to the construction of the start and end points of our connecting segment, the curvature is zero and the tangents of the connecting curve and the circle are orthogonal. Without loss of generality, we assume that the heading and the position at the beginning of the segment is zero. For heading δ after the transition follows

$$\sigma(\delta) = \frac{\pi\left[\cos\frac{\delta}{2} C\left(\sqrt{\frac{\delta}{\pi}}\right) + \sin\frac{\delta}{2} S\left(\sqrt{\frac{\delta}{\pi}}\right)\right]^2}{\sin^2\left(\frac{\delta}{2}\right)} \quad (2)$$

and a length $L(\delta) = 2\sqrt{\frac{\delta}{\sigma}}$ of the clothoid segment. We can construct the connecting segment from two clothoid segments, one starting with curvature zero at the start point to an intermediate point with heading $\frac{\delta}{2}$ and a segment mirrored at the end point of the first segment. Fig. 2 illustrates the segment construction. Finally, the resulting normalized connecting segment is scaled and projected back into the 3D space. To simplify the calculations, we do not restrict the maximum curvature. This ensures that a continuous solution can always be found even if locally not dynamically feasible. We leave finding a globally feasible solution to the subsequent optimization stage.

To allow for arbitrary spaced sampling, we fit cubic splines to the resulting path consisting of straight line segments and clothoid transition segments. In contrast to fitting splines to the initial path containing only straight line segments, we

achieve a much closer approximation of the path. Thus, the spline-based path is always collision-free if the planned path is collision-free.

V. Local Multiresolution Optimization

To react on deviations from the planned trajectory and to avoid obstacles perceived with onboard sensors, frequent replanning is inevitable. Optimizing a high-dimensional trajectory with a time resolution of $10\,\text{ms}$ is prohibitively slow. Nevertheless, the high time resolution at fixed duration is only required for the prediction horizon of the MAV controller. If we perform replanning sufficiently fast, the remainder of the trajectory can be represented at a lower time resolution, even with varying durations of trajectory points. We employ a local multiresolution time discretization—extending ideas from our prior work [20]. The prediction horizon is represented at a constant high resolution, after that the resolution decreases linearly. This resembles the uncertainty in the trajectory execution and perception in the future. The duration of trajectory point i is $d(i) = \Delta t$ for $i \leq i_{\text{fix}}$ and $d(i) = (1 + c \cdot (i - i_{\text{fix}})) \cdot \Delta t$ for $i > i_{\text{fix}}$ with multiresolution factor $c = 0.1$ in our implementation. This results in an overall duration from the trajectory start to point i of $D(i) = i\Delta t$ for $i \leq i_{\text{fix}}$ and

$$D(i) = i_{\text{fix}}\Delta t + \left(\Delta i + \sum_{k=1}^{\Delta i} ck\right)\Delta t \tag{3}$$

$$= i_{\text{fix}}\Delta t + \left(\Delta i + c\frac{\Delta i^2 + \Delta i}{2}\right)\Delta t \tag{4}$$

with $\Delta i = i - i_{\text{fix}}$ for $i > i_{\text{fix}}$. We set i_{fix} to 10, hence replanning has to be performed in $100\,\text{ms}$.

To calculate the derivatives for our trajectory given an arbitrary time discretization, we employ finite differencing [21]. Due to the non-equal time difference between consecutive trajectory points, it is necessary to compute a differencing filter per trajectory point. Nevertheless, the time discretization is fixed over the whole optimization process, thus these filters can be precomputed. To allow for numerical differentiation of the first trajectory points, an additional padding of six time steps of fixed resolution is added at the beginning and end of the trajectory. The padding is fixed and not altered during the optimization process. It is initialized with the start and goal configurations, respectively.

During execution of the trajectory, we shift the trajectory padding corresponding to the elapsed time. Hence, the trajectory padding always contains the past six trajectory states representing the dynamic state in the past that led to the current state. Consequently, the trajectory optimization finds feasible followup trajectories implicitly. The remaining trajectory is mapped to the new time parametrization by means of linear interpolation between trajectory points. After this reparameterization, some subsequent optimization steps are necessary to find the new local optimal solution, even if the MAV was not disturbed. In our implementation, we have restricted the reoptimization iterations to one fifth of the initial optimization iterations. As the trajectory points in the padding are not differentiable with the finite differences method, their derivatives have to be represented explicitly. Thus, our intermediate trajectories are 12+1 dimensional, containing 4D poses, velocities, accelerations plus duration.

If complete replanning is necessary during the trajectory execution due to topological changes in the environment that cannot be solved by reoptimization, the initialization steps including allocentric planning have to be repeated. In contrast to the initial planning, the trajectory padding of the sampled trajectory is now filled with a future part of the currently executed trajectory to take the dynamic state of the MAV into account for the followup trajectory.

VI. Evaluation

We evaluate our approach in simulation employing the RotorS simulator [22] and perform a feasibility experiment on our MAV in a motion capture system. Fig. 3 shows the reduction of the maximum accelerations at transitions between consecutive path segments. The initial planned path has discontinuities in the derivative of the trajectory causing large accelerations for single time steps. In contrast, our approach—a combination of continuous curvature transitions with spline interpolation—reduces the maximum acceleration at transition points significantly. Spline interpolation alone—without clothoid segments—has the potential to reduce the accelerations further by dropping guaranteed obstacle-freeness. Our approach preserves obstacle-freeness by inserting local changes into the plan in an obstacle-free volume. We depict the acceleration reduction factors in dependence of the transition angle and radius of the free space spheres in Fig. 4. We generated $10\,\text{m}$ long trajectories with one transition between line segments in the middle with uniform time discretization. Resulting acceleration reductions are reported for free space from $5\,\text{cm}$ up to $5\,\text{m}$ and typical angles of $45°$, $90°$, and $135°$. The initial accelerations already are an order of magnitude smaller within an free space volume of $1\,\text{m}$ with our approach for typical heading changes. Only the angle of $135°$ shows less reduction, because the maximum curvature for large heading changes still causes large accelerations when performed in a small radius. Nevertheless, such large heading changes are very uncommon with a grid-based planner in real world scenarios.

In simulation, the MAV followed trajectories around a power plant that is part of the RotorS simulator. Fig. 1 shows an example trajectory. We defined a path defined by 12 via-points that constrain the optimization topological to a trajectory traversing eight apertures in the building. Our preprocessing calculates free-space bounding spheres at the via-points and inserts continuous curvature transition segments. Finally, the path is optimized and executed with frequent reoptimization.

Fig. 5 shows the result of frequent reoptimization during trajectory execution. When using different multiresolution factors c, the resulting optimized trajectories diverge with increasing duration from the start pose. For our evaluation, we reoptimized the trajectory every ten time steps, i.e., every $100\,\text{ms}$, during execution. Plots of the actual flown

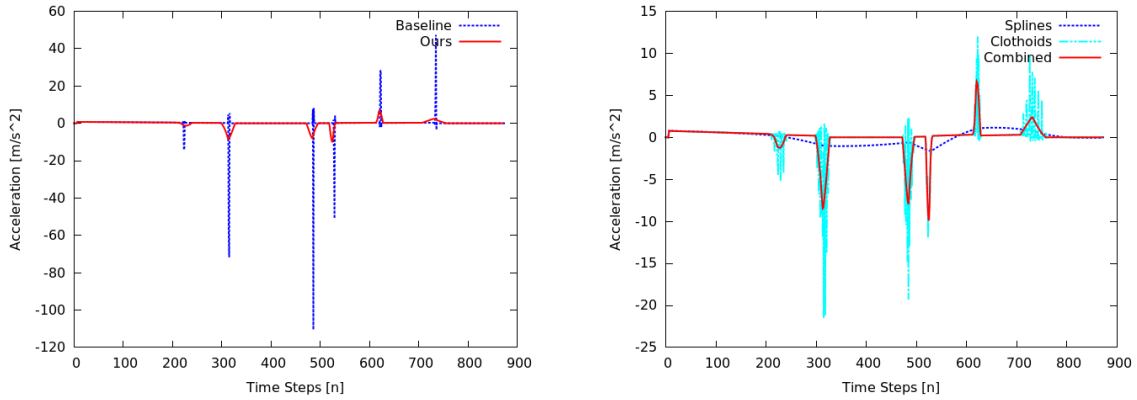

Fig. 3. Accelerations of initial trajectory. Left: Straight line trajectories have discontinuities causing large accelerations at the transition points (Baseline, planned path from A* planner). Our approach—a combination of continuous curvature transitions with spline interpolation—reduces these accelerations much closer to feasible accelerations (approx. $3\,\mathrm{m\,s^{-2}}$ for our MAV). Better initial guesses for the trajectory lead to faster convergence. Right: Spline interpolation combined with clothoid transitions further mitigate effects caused by discretization. Splines fitted to the initial trajectory result in even smaller accelerations, but cannot assure obstacle-freeness of the resulting path. Our approach alters the path only locally in obstacle-free volumes.

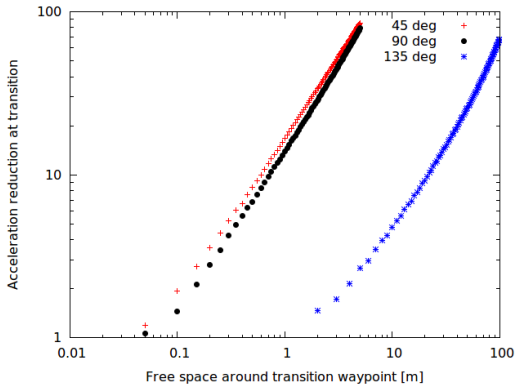

Fig. 4. Acceleration reduction with continuous curvature transition segments. Depicted is the reduction factor $a_{\max}^{\text{baseline}}/a_{\max}^{\text{ours}}$ of the maximum accelerations encountered with the baseline trajectory—planned with the A* planner—and ours on a double logarithmic scale. An order of magnitude of reduction can already be observed at less than one meter of free space with smaller transition angles. A transition with an angle of 135° still causes large accelerations when generated in a small free space sphere due to the high necessary maximum curvature. In non-artificial scenarios angles beyond 90° are uncommon.

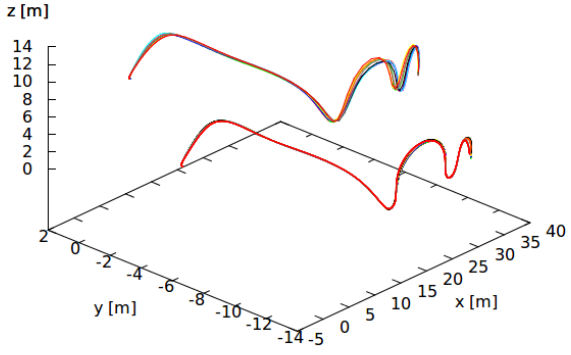

Fig. 5. Comparison of initially planned and executed multiresolution trajectories. The top bundle of trajectories depicts the multiresolution trajectories from start (left) to goal (right) for multiresolution factors c from 0.1 to 1. The trajectories diverge with increasing duration caused by the different discretizations. Nevertheless, the bottom bundle shows that the resulting trajectories when executed are very similar due to frequent reoptimization during execution. For better visibility the two bundles are plotted with an artificial offset.

trajectories are depicted in the bottom part of Fig. 5. These trajectories are very similar to each other, hence frequent reoptimization mitigates the effects of larger multiresolution factors facilitating faster optimizer iterations.

For initial optimization, we employ 500 iteration steps. Fig. 6 shows the duration of these initial optimizations, depending on the selected multiresolution factor and the uniform discretization of the trajectory. All timings are measured on a computer with Intel Core i7 940 CPU.

Real Robot Experiments To test the applicability of our approach, we performed proof-of-concept real robot experiments. We followed trajectories generated by our approach with uniform timing trajectory optimization with Δt = 10 ms with an AscTec Neo hexacopter. State feedback was provided by a motion capture system. Fig. 7 shows the results of one exemplary trajectory execution. The shape and dynamics of the reference trajectories are well matched by the MAV, showing the applicability of our approach to robot navigation.

VII. Conclusion

We have developed a method to facilitate fast trajectory optimization. We modify trajectories generated by a complete allocentric path planner locally—while keeping their guaranteed obstacle-freeness—with continuous curvature transition segments based on clothoids. This significantly reduces acceleration peaks and thus expedites the convergence of trajectories to a local optimal solution. Trajectory optimization itself is sped up by means of a local multiresolution discretization of the time dimension along trajectories. This makes frequent reoptimization feasible and allows to plan trajectories with the same time discretization as the low-level controller generating attitude and thrust setpoints for the MAV.

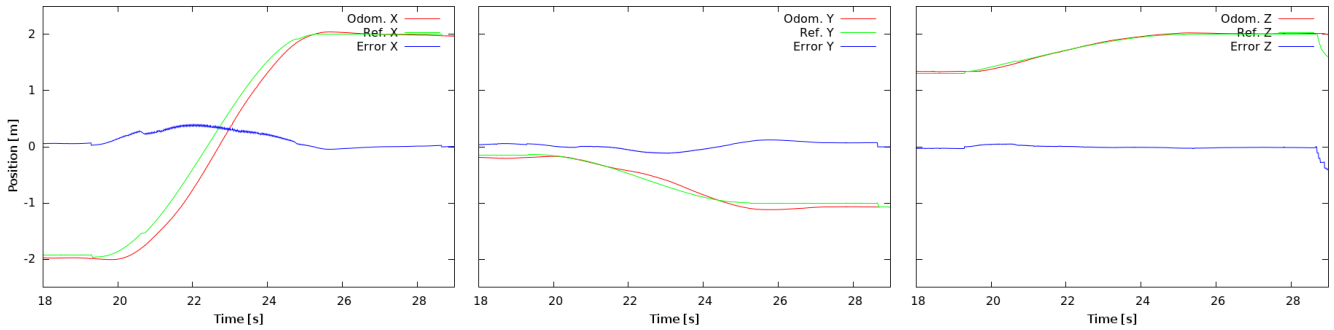

Fig. 7. Trajectory following with AscTec Neo MAV. We tested the dynamic feasibility of the generated trajectory for direct execution on an MAV while flying in a motion capture system. The commanded trajectories could be followed with only a small time delay. Thus, the motion dynamics model employed during optimization resembles the real MAV flight dynamics. The green graph depicts the reference trajectory, the red graph pose measurements from the motion capture system. Blue depicts the trajectory error.

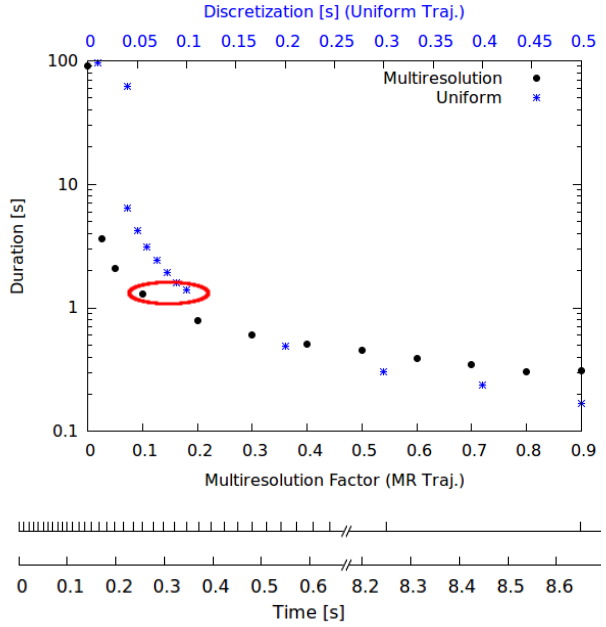

Fig. 6. Top: Optimization duration with and without multiresolution. Black dots depict the duration of 500 optimizer iterations—the number of iterations we use for the initial optimization—depending on the multiresolution factor c in Eq. 3. Blue stars depict the duration of non-multiresolution optimization depending on the uniform discretization of the trajectory. Whereas the uniform time resolution has to be reduced by at least an order of magnitude to achieve acceptable optimization speed, introducing multiresolution allows for real-time replanning with high resolution in the controller prediction horizon. In our implementation we use a multiresolution factor of 0.1. The optimization requires approximately the same amount of time as a uniform discretization of 0.1 s (circled red). Bottom: Comparison of the time discretization for the circled cases. The bottom line depicts the uniform time discretization which is an order of magnitude coarser than needed by low-level control. The top line depicts the multiresolution discretization which is much finer at the beginning and gets coarser in the future.

References

[1] M. W. Mueller, M. Hehn, and R. D'Andrea, "A computationally efficient motion primitive for quadrocopter trajectory generation," *IEEE Trans. on Robotics*, 2015.

[2] C. Richter, A. Bry, and N. Roy, "Polynomial trajectory planning for quadrotor flight," in *Int. Conf. on Robotics and Automation*, 2013.

[3] M. Zucker, N. Ratliff, A. Dragan, M. Pivtoraiko, M. Klingensmith, C. Dellin, J. A. D. Bagnell, and S. Srinivasa, "Chomp: Covariant hamiltonian optimization for motion planning," *Int. J. of Robotics Research*, vol. 32, pp. 1164–1193, 2013.

[4] L. Heng, D. Honegger, G. H. Lee, L. Meier, P. Tanskanen, F. Fraundorfer, and M. Pollefeys, "Autonomous visual mapping and exploration with a micro aerial vehicle," *J. of Field Robotics*, vol. 31, no. 4, pp. 654–675, 2014.

[5] K. Schmid, P. Lutz, T. Tomic, E. Mair, and H. Hirschmüller, "Autonomous vision-based micro air vehicle for indoor and outdoor navigation," *J. of Field Robotics*, vol. 31, no. 4, pp. 537–570, 2014.

[6] M. W. Achtelik, S. Lynen, S. Weiss, M. Chli, and R. Siegwart, "Motion- and uncertainty-aware path planning for micro aerial vehicles," *J. of Field Robotics*, vol. 31, no. 4, pp. 676–698, 2014.

[7] B. MacAllister, J. Butzke, A. Kushleyev, H. Pandey, and M. Likhachev, "Path planning for non-circular micro aerial vehicles in constrained environments," in *Int. Conf. on Robotics and Automation*, 2013.

[8] H. Cover, S. Choudhury, S. Scherer, and S. Singh, "Sparse tangential network (SPARTAN): Motion planning for micro aerial vehicles," in *Int. Conf. on Robotics and Automation*, 2013.

[9] I. Şucan and L. Kavraki, "Kinodynamic motion planning by interior-exterior cell exploration," *Algorithmic Foundation of Robotics VIII*, vol. 57, pp. 449–464, 2009.

[10] S. M. LaValle and J. J. Kuffner, "Randomized kinodynamic planning," *Int. J. of Robotics Research*, vol. 20, no. 5, pp. 378–400, 2001.

[11] M. Kalakrishnan, S. Chitta, E. Theodorou, P. Pastor, and S. Schaal, "Stomp: Stochastic trajectory optimization for motion planning," in *Int. Conf. on Robotics and Automation*, 2011.

[12] C. Park, J. Pan, and D. Manocha, "Itomp: Incremental trajectory optimization for real-time replanning in dynamic environments," in *Int. Conf. on Automated Planning and Scheduling (ICAPS)*, 2012.

[13] F. Augugliaro, A. P. Schoellig, and R. D'Andrea, "Generation of collision-free trajectories for a quadrocopter fleet: A sequential convex programming approach," in *IROS*, 2012.

[14] K. Bipin, V. Duggal, and K. M. Krishna, "Autonomous navigation of generic quadrocopter with minimum time trajectory planning and control," in *Int. Conf on Vehicular Electronics and Safety*, 2014.

[15] H. Andreasson, J. Saarinen, M. Cirillo, T. Stoyanov, and A. J. Lilienthal, "Drive the drive: From discrete motion plans to smooth drivable trajectories," *Robotics*, vol. 3, no. 4, pp. 400–416, 2014.

[16] A. Majumdar and R. Tedrake, "Funnel Libraries for Real-Time Robust Feedback Motion Planning," *ArXiv e-prints*, Jan. 2016.

[17] K. He, E. Martin, and M. Zucker, "Multigrid CHOMP with local smoothing," in *Int. Conf. on Humanoid Robots*, 2013.

[18] M. Nieuwenhuisen and S. Behnke, "3D planning and trajectory optimization for real-time generation of smooth MAV trajectories," in *European Conf. on Mobile Robots (ECMR)*, 2015.

[19] T. Fraichard and A. Scheuer, "From Reeds and Shepp's to continuous-curvature paths," *IEEE Trans. on Robotics*, vol. 20, no. 6, pp. 1025–1035, 2004.

[20] R. Steffens, M. Nieuwenhuisen, and S. Behnke, "Continuous motion planning for service robots with multiresolution in time," in *Int. Conf. on Intelligent Autonomous Systems (IAS)*, 2014.

[21] B. Fornberg, "Generation of finite difference formulas on arbitrarily spaced grids," *Mathematics of computation*, vol. 51, no. 184, pp. 699–706, 1988.

[22] F. Furrer, M. Burri, M. Achtelik, and R. Siegwart, "RotorS–a modular gazebo MAV simulator framework," in *Robot Operating System (ROS): The Complete Reference*, A. Koubaa, Ed., 2016, vol. 1, ch. 23, pp. 595–625.