European Conference on Mobile Robots (ECMR), Lincoln, UK, September 2015.

3D Planning and Trajectory Optimization for Real-time Generation of Smooth MAV Trajectories

Matthias Nieuwenhuisen and Sven Behnke
Autonomous Intelligent Systems Group, Institute for Computer Science VI
University of Bonn, Germany

***Abstract*—Complex indoor and outdoor missions for autonomous micro aerial vehicles (MAV) constitute a demand for 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 utilizes coarse grid-based path planning with an approximate model of flight dynamics to determine collision-free trajectories. To account for the flight dynamics and to mitigate discretization effects, these trajectories are further optimized with a gradient-based motion optimization method. We evaluate our method on an outdoor map with buildings and report trajectory costs and runtime results.**

I. INTRODUCTION

In recent years, micro aerial vehicles (MAVs) became increasingly popular for inspection and surveillance tasks. With MAVs it is possible to reach otherwise inaccessible areas with low effort, e.g., it is possible to inspect structures in higher altitudes without the need for scaffolds or climbers. Nevertheless, at the moment most of the MAVs are remotely controlled or navigate to fixed GPS waypoints and hold the position. This restricts the applications to well observable obstacle-free areas in the line of sight of an operator. Flying in more challenging 3D environments, like low altitude flight in outdoor environments with vegetation or indoor environments, demands a higher level of autonomy.

Especially on larger sites, a constant connection to the MAV may not be maintainable. Also, passages may be narrow and surrounding environmental structures may be hard to perceive for a human operator. In order to safely navigate in such environments, an alternative is to have an autonomous MAV that can on its own—and without interaction with the operator—solve well-defined sub-tasks, i.e., autonomously approach multiple view-poses and collect (and/or transmit) sensor information. For the autonomous operation of MAVs, a key prerequisite is the planning of collision-free trajectories.

To quickly react on changing environments, planning times need to be low to allow for frequent replanning. We achieve this by employing a multi-layer approach to navigation: from slower deliberative to fast reactive layers, including mission planning, global and local path planning, fast local obstacle avoidance, and robust motion controllers [1].

In contrast to fixed-wing MAVs, multirotors are capable of flying omnidirectionally and can stop or change direction within a short time horizon. Hence, when flying with low velocity, the dynamics of a multirotor MAV can be neglected on higher planning layers. When aiming at shorter mission execution times, higher average velocities are desirable, though.

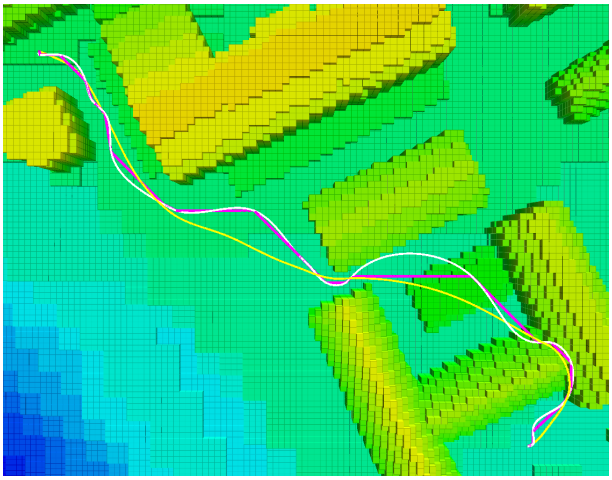

Fig. 1: Exact planning of smooth trajectories with velocity and acceleration dimensions for MAVs is often not feasible fast enough needed for frequent replanning. We plan coarse 3D trajectories (pink), incorporate simple assumptions about the MAV's dynamics (white), and optimize this initial guess quickly to obtain a least cost kinodynamic trajectory (yellow) w.r.t. distance to obstacles and control costs.

To achieve these, we extend our previous work on allocentric path planning by optimizing flight trajectories with a simple MAV dynamic model.

In this work, we employ a static environment model acquired by SLAM from prior flights [2] or derived from other sources, e.g., building models. 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 the suboptimal initialization by approximating optimal velocities and accelerations with a simple MAV dynamics model and by smoothing the input trajectory to a trajectory requiring less control effort. With these fast preprocessing steps, we achieve a combined planning and optimization time of approximately 1 s for allocentric trajectory generation, for the example depicted in Fig. 1. We evaluate the effects of the intermediate processing steps in a simulated large-scale outdoor environment.

II. Related Work

The application of MAVs varies especially in the level of autonomy—ranging from basic hovering and position holding [4] over trajectory tracking and waypoint navigation [5] to fully autonomous navigation [6]. Particularly important for fully autonomous operation is the ability to perceive obstacles and to avoid collisions. Obstacle avoidance is often neglected, e.g., by flying in a sufficient height when autonomously flying between waypoints.

A good survey on approaches to motion planning for MAVs is given in [7]. Due to the limited computational power onboard the MAV, especially low computational costs are crucial for the applicability of these methods. To meet real-time demands, layered planning approaches are often used.

Israelsen et al. [8] present an approach to local collision avoidance that works without global localization and can aid a human operator to navigate safely in the vicinity of obstacles. Our work extends the safety layer by a deliberative planning layer based on local maps and commands.

Heng et al. [9] 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. [10] 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. [11] 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. [12]. 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. [13] use a search-based method as well.

A two-level approach to collision-free navigation, using artificial potential fields on the lower layer is proposed by Ok et al. [14]. Similar to our work, completeness of the path planner is guaranteed by an allocentric layer. Andert et al. [15] use a three-level hierarchical behavior control algorithm to fly a helicopter through a gate. Whalley et al. [16] employ five navigation layers to fly 230 km with a helicopter. Obstacles are detected and avoided with an onboard laser scanner. While in their work sensing and consequently planning is limited to a narrow FoV in flight direction, we employ full 3D planning, including flying sideways and backwards. Johnson et al. [17] use reactive obstacle avoidance on a small helicopter for velocities up to 12 m/s.

To plan high-dimensional trajectories, often sampling-based planners are employed, including KPIECE [18] and randomized kinodynamic planning [19]. Implementations for many sampling-based planners are provided in the Open Motion Planning Library (OMPL) [20]. 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. [21]. STOMP combines the advantages of CHOMP with a stochastic approach. In contrast to CHOMP, it is no longer required to use cost functions for which gradients are available, while the performance stays comparable. This allows to include costs with regard to, for instance, general constraints or motor torques. Another algorithm derived from CHOMP is ITOMP, an incremental trajectory optimization algorithm for real-time replanning in dynamic environments [22]. 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. [23] compute collision-free trajectories for multiple MAVs simultaneously. Other obstacles than the MAVs are not considered here. Similar to our approach, Richter et al. [24] 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. [25]). Collisions are avoided by intermediate waypoints from a high-level planner and are not explicitly considered in the optimization process.

III. Trajectory Optimization

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.

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 $x_0 = (x_0, y_0, z_0, \theta_0)^\top, x_N = (x_N, y_N, z_N, \theta_N)^\top \in \mathbb{R}^4$. 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 with fixed duration Δt. 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

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

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]. If a gradient for the used cost function cannot be computed, an alternative is to optimize the trajectory w.r.t. to the cost function $q(\tilde{\Theta})$ with

$\tilde{\Theta} = \mathcal{N}(\Theta, \Sigma)$ being a noisy state parameter vector with mean Θ and covariance Σ by means of a stochastic optimizer [21].

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 fixed 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 minimum trajectories.

IV. Model-based Initialization

In order to generate a collision-free flight trajectory between mission points, we employ a three-step approach. First, we plan a coarse obstacle-free 3D path, second, we process this plan to fill missing dimensions (i.e., yaw rotation, velocities) with an initial guess as close as possible to the expected final result, and third, we optimize this initialization trajectory w.r.t. control and obstacle costs to obtain a smooth trajectory that can be followed with low control effort.

To efficiently obtain obstacle costs during planning, we calculate a distance field [26] with a 20 cm resolution from our static environment model. We propagate distances up to a maximum distance where obstacle costs are zero. Our distance-dependent obstacle costs are modeled as a piecewise linear function, with decreasing costs up to a maximum distance from obstacles of 4 m.

Initial path planning is performed in a 3D grid employing the A* algorithm. As the convex hull of our robot is approximately a cylinder along the z-axis, we neglect the robot orientation at this point. In later steps, we interpolate between start and goal orientation along the trajectory. We simplify the resulting plans to reduce discretization effects. These plans can already be executed on the MAV with a simple position controller, but the MAV has to lower its velocity at every waypoint.

The hard transitions between consecutive segments of the plan cause large acceleration changes, exceeding the MAV dynamic limits at high velocities. Due to the low angular resolution of the grid-based planner, the path is discontinuous at the connection of consecutive plan segments. Hence, the derivatives calculated by the trajectory optimizer have spikes with high values. During the optimization process, the convergence at these points is poor which increases the overall optimization time. We employ cubic spline interpolation to acquire smoother trajectories. The sampling points are the endpoints of the segments from the simplified path at the timesteps from our motion model. Splines mitigate the discretization effects leading to lower accelerations. Fig. 2 shows

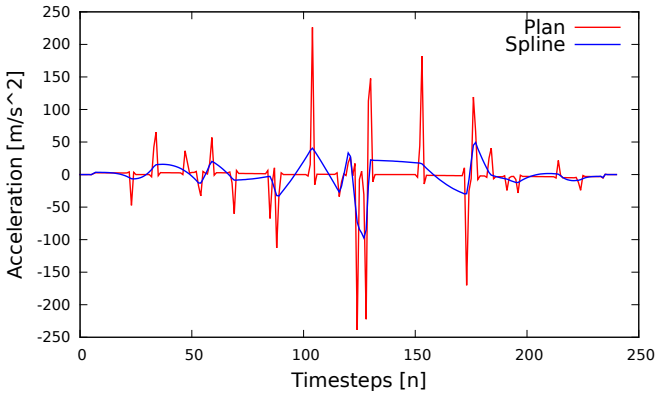

Fig. 2: Accelerations of the trajectory before optimization. Initialization with the path from a grid-based planner yields high accelerations at the connection points between plan intervals. Spline-interpolation of the path reduces these spikes yielding faster convergence.

the necessary accelerations to follow the planned and spline-based trajectories. Nevertheless, splines can overshoot and cause collisions without further processing. Furthermore, the trajectories tend to oscillate. The derivatives at the sampling points are omitted. Velocities and accelerations are calculated after the interpolation. Even though the spline-based trajectories are smooth, acceleration and velocity limits can still be violated without further optimization by non-optimal timings and large curvature necessary to pass the planned waypoints.

To use this plan as initialization for the trajectory optimizer, we have to rediscretize the plan to match the fixed-duration timesteps of the parameter vector. To get an easy to compute closed form solution for our discretization, we assume that the MAV starts with a maximum acceleration $a(0) = a_{max}$, stops with a maximum deceleration $a(T) = -a_{max}$ at the end of the trajectory, and a linear transition between these states. With an estimated flight duration of T for the whole trajectory, we can derive a simple motion model of the MAV for acceleration $a(t)$, velocity $v(t)$, and position $x(t)$

$$a(t) = -2\frac{a_{max}}{T}t + a_{max}, \quad (1)$$

$$\int a(t)\mathrm{d}t = v(t) = -\frac{a_{max}}{T}t^2 + a_{max}t, \quad (2)$$

$$\iint a(t)\mathrm{d}t = x(t) = -\frac{a_{max}}{3T}t^3 + \frac{1}{2}a_{max}t^2, \quad (3)$$

at time $t \in [0, T]$.

With $x(T) = L$, given a total length L of the planned path, and (3) we can calculate the estimated flight duration T as

$$L = -\frac{a_{max}}{3T}T^3 + \frac{1}{2}a_{max}T^2 \quad \Rightarrow \quad T = \sqrt{\frac{6L}{a_{max}}}.$$

With $T = (N+1)\Delta t$, we get the necessary number of time steps N for our trajectory discretization.

A uniform discretization of the planned path into these N timesteps can serve as an input to the optimizer, but the derivatives (constant velocity, zero acceleration) are far from optimal. Hence, a large amount of optimization effort is spent on optimizing the timing of the trajectory.

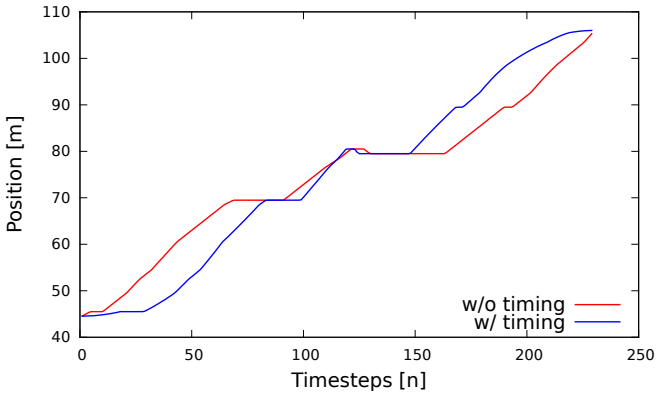

Fig. 3: Uniform plan discretization (red) vs. discretization according to motion model (blue) of the MAV's trajectory in the x-coordinate. Discretizing according to a motion model facilitates faster convergence of the trajectory optimizer.

The position $x(t) \in [0, L]$ is the part of the planned path that has been traversed until time t. This can be used to rediscretize the path with a better initial guess about velocities and accelerations that reduce the control costs over the complete trajectory. Fig. 3 shows the effect of the timing-based discretization on the initial parameter vector. The velocities at the start and goal are lower and the velocity in the middle of the trajectory is higher.

Fig. 4 shows a comparison of the position trajectories for (x, y, z) after initial planning, spline interpolation and optimization.

V. Frequent Optimization

To cope with newly perceived obstacles and deviations from the planned trajectory during flight, we optimize the trajectory frequently. To calculate the derivatives of the trajectory points by finite differencing [27], the parameter vectors Θ^d have a padding of 6 parameters at start and end with fixed values not changed during optimization. In the first run, the paddings are filled with start and goal configuration of the trajectory.

For frequent optimization, we increase the start index of the parameters to optimize with the elapsed execution time and move the fixed length start padding window forward with this index. As result, the padding contains the next future timesteps of the trajectory which are not altered in the next optimization iteration and can be executed by the MAV during processing of the new trajectory. This ensures feasible dynamic transitions from the currently followed trajectory to the new optimization result.

As the optimizer is initialized with the already optimized trajectory from the previous iteration, the initialization steps necessary for the first trajectory can be omitted and the trajectory converges fast to a new local optimum in cases of small changes in the environment, e.g., small obstacles or slow dynamic obstacles. The optimization cannot leave local optima if another trajectory becomes closer to a global optimum. This can happen by newly perceived obstacles blocking or influencing the old locally optimal trajectory. To avoid this, we perform global replanning from points on the trajectory in the more distant future to the goal and newly initialize and optimize the remaining part of the trajectory, similar to the initial trajectory planning.

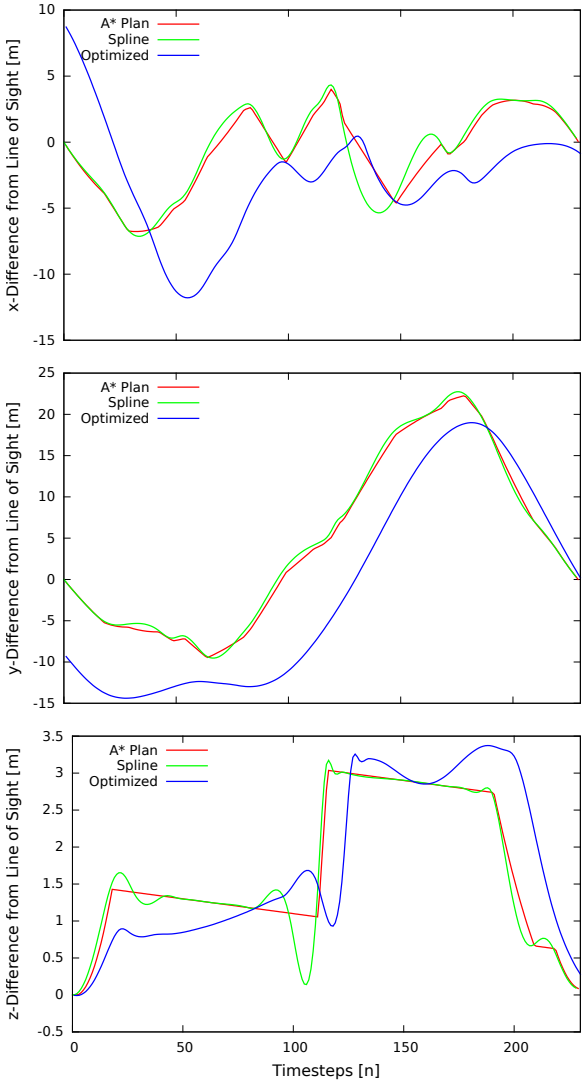

Fig. 4: Comparison of trajectories for individual position dimensions at different stages of optimization. Planned paths (red) with applied timing correction require still large accelerations when the movement direction changes. Spline interpolation (green) mitigates these effects, but tend to overshooting and are bound to the initial plans sampling points. The optimized trajectories (blue) are much smoother and reduce the necessary control effort.

To perceive obstacles during flight, our MAV employs a rotating 3D laser scanner [1]. Our scanner measures 1080 distances per scan line at 40 Hz. The maximum time for incorporating a single scan line into the distance field is 29 ms, exceeding the time window by 4 ms. However, when incorporating complete 3D scans of an cluttered environment with obstacles in any direction into an empty distance field, the updates are possible in real-time on average. On average, the time per scan line for the first complete 3D scan of an environment—a half rotation of the laser scanner, yielding 20 scan lines—is approximately 10 ms. With only small changes in the environment and no movement of the MAV, distance

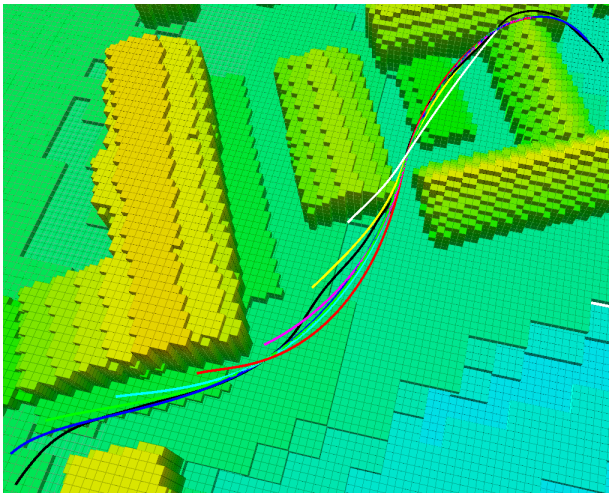

Fig. 5: We employ frequent optimization to recover from disturbances during trajectory execution. In this example the MAV follows an initial trajectory from start to goal (black). Gusts of wind push it away from the trajectory, the colored trajectories depict the newly optimized trajectories after every gust of wind.

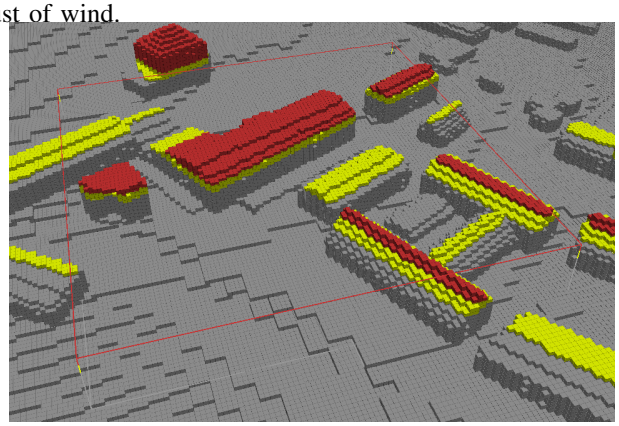

Fig. 6: OctoMap of the evaluation area. We have restricted the maximum allowed altitude for our experiments to approx. 10 m over ground, otherwise flying at higher altitudes always yields shortest paths. Yellow: Obstacles influencing the MAV at 10 m altitude. Red: Obstacles the MAV cannot overfly.

field updates are performed in 0.2-5 ms per scan line.

To evaluate the capability to recover from disturbances during trajectory execution, we simulate strong gusts of wind while following an initial trajectory. Fig. 5 shows the resulting trajectories in an experiment where the MAV is pushed away 3 m from its current position every second until passing a building higher than the flight altitude. As the MAV dynamic state is not altered and thus the motion direction cannot be changed immediately, we move the initial part of the old trajectory to the deviated current MAV pose. We distribute the trajectory error over the remaining part of the trajectory. This part is then used as initialization for optimization yielding locally optimal trajectories to the goal.

VI. Evaluation

We evaluate our approach on an outdoor map containing buildings from a farm area, depicted in Fig. 6. In this environment, shortest paths are often direct connections at a certain height. To avoid this simple solution, we restrict the allowed flight altitude to a fixed absolute height. Depending on the terrain elevation, this limit is 10-14 m above ground-level. Some buildings are higher than this allowed altitude and have to be surrounded by the MAV. The path planning grid has a size of $100\times100\times14$ m and a cell size of 1 m. Our distance field is 3 m larger in every dimension to allow for correct gradient calculations and has a resolution of 20 cm. The higher resolution of the distance field compared to the planner grid is exploited in the following optimization step. The allowed minimum distance to obstacles is 2 m, the maximum distance influenced by an obstacle is 4 m. The generation of the initial distance field from an OctoMap takes 6.1 s. All timings are evaluated on a single core of our MAV [28] onboard computer equipped with an Intel Core i7-3820QM CPU running at 2.70 GHz.

In the first experiment, we plan a path of 229 m for further optimization. A valid path is found in 0.46 s. Our second step, the calculation of timings and spline-based-trajectories runs in under 1 ms. Fig. 7 shows the convergence of the trajectory optimizer in our four evaluation cases. 1) uniform sampling of the planned path as initialization for the optimizer, 2) sampling of the planned path according to a motion model, 3) spline interpolation with uniform sampling, 4) combining spline interpolation and motion model. The costs of a trajectory are a sum of state and control costs. The control costs penalize the change in control input, i.e., minimize the jerk of the trajectory. State costs incorporate obstacle costs, accelerations, and velocity of the MAV. Clearly, the spline-based initializations start with much lower trajectory costs and converge faster to the local optimum. The effect of the motion model-based timings are visible in the plan-based initializations. The model-based initialization combined with splines is not visible at the beginning of the optimization. This can be explained by a larger overshoot causing higher velocities and obstacle costs in parts of the initial trajectory. We depict the control cost part without state costs in Fig. 8. Here, the initialization with motion model-based timings is better in the plan and spline case. We achieve 25% and 34% lower control costs in the beginning by using improved timings and approximately 75% lower initial costs by using splines. In combination, the initial control costs can be reduced by 77%. In normal operation of we stop the optimization after 500 iterations. The optimization process takes 0.96 s for trajectory points with a Δt of 0.05 s.

Fig. 1 shows resulting trajectories from each of the processing steps, i.e., a planned path, a spline interpolation of this path with motion model-based timings, and the resulting trajectory after 500 optimizer iterations.

In the second experiment, we generate trajectories for pairs of obstacle-free start and goal poses uniformly distributed over the evaluation area. We omit trajectories between poses with less than 70 m distance. This results in 1,216 trajectories with an average length of 101 m. The shortest planned path was about 72 m, the longest trajectory was about 155 m. We stop the optimization after 500 iterations and evaluate the trajectory cost reduction with our proposed initializations during optimization. We calculate the cost reduction r_i after optimizer iteration i as $r_i = \left(1 - c_i / c_i^{base}\right) \cdot 100\%$. Here, c_i^{base} is the average cost of a trajectory after iteration i with the baseline algorithm and c_i is the average cost with the evaluated

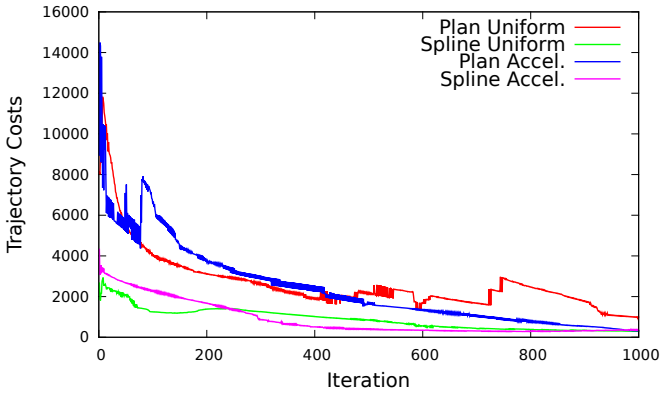

Fig. 7: Trajectory costs (state and control costs) per iteration of the optimizer. Whereas the optimizer converges to nearly the same value for all initializations, spline-based initializations (green/pink) reach a low value much faster. Also the combination of spline-based initialization with non-uniform accelerations (pink) reduces the convergence time.

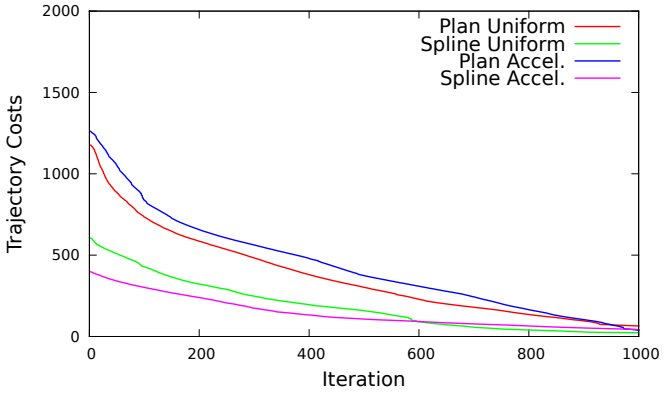

Fig. 8: Summed control costs per iteration of the optimizer. After initialization the overall control costs of the trajectory could be reduced by 25 - 34% by employing better timings and by approx. 75% by using splines in this example.

initialization. Fig. 9 shows the trajectory cost at each iteration compared to the baseline, i.e., direct initialization with the plan from the grid-based planner. With enough iterations, the cost reduction converges to zero as the optimization initialized with the baseline approach will finally converge to the local optimum, but this makes frequent planning unfeasible. Especially spline-based initialization reduces the initial cost drastically. Combined with the motion model-based timing correction, after 500 iterations the trajectory is still less costly than without. By means of spline interpolation, the initial costs can be significantly reduced. This leads to faster convergence resulting in 20-45% less costly trajectories after 250 iterations and 9-24% less costly trajectories after 500 iterations, compared to the baseline.

Employing motion model-based timings reduces the cost at some iterations when directly applied to a planned path. But higher velocities and accelerations in other iterations can lead to much higher control costs in cases where connections between plan segments have to be traversed with high speeds. This results in low improvements or even negative effects. A positive effect can be observed in the long run, after the initial plan has been smoothed enough by optimization.

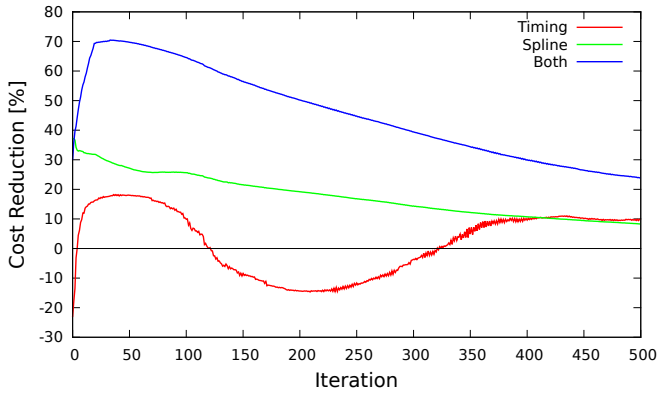

Fig. 9: Reduction of trajectory costs compared to baseline (initialization with A* planned path) in each optimizer iteration. The results are an average over 1,216 trajectories.

TABLE I: Runtimes of planning and optimization averaged over 1,216 trajectories.

	Mean (s)	Std Dev. (s)	Max. (s)
Path Planning	0.07	0.03	0.29
Optimization	0.58	0.13	1.16
Total	0.64	0.14	1.45

Tab. I shows the average and maximum runtimes of the planning and trajectory optimization. In 98.8% of the optimization runs, the summed planning and optimization times are below 1 s. Some more complex trajectories take longer planning and optimization time (maximum 1.45 s), yielding an average total optimization time of 0.64 s.

We evaluated the frequent reoptimization by simulating strong gusts of wind while the MAV follows a trajectory. As baseline we move an initial fixed part of the trajectory to the new MAV position and perform complete replanning and optimization from the endpoint of that fixed trajectory part to the goal. The fixed part is the part the MAV will follow during replanning, due to its current dynamic state. Fig. 10 shows the cost reduction of the trajectory during initial optimization and while repairing the trajectory after two gusts of wind, each pushing the MAV away 4.25 m. Reoptimization yields a close-to-optimal cost trajectory in less than 100 iterations. In contrast the complete replanning needs about 500 iterations. To evaluate the overall compute time we simulated MAV flights, disturbed every second by strong gusts of wind. On average the reoptimization finished in 18.3% of the time complete replanning took. The maximum was 28.3% and the minimum 14.8%. Without disturbances the trajectory improves with every reoptimization step.

VII. Conclusion

In this paper, we presented an approach to speed up trajectory generation for MAVs based on a grid-based path planner and the trajectory optimizer CHOMP. This allows for frequent replanning during mission execution. This work extends our prior work on fast MAV planning by taking dynamic constraints of the robot into account, facilitating higher possible execution speed. The optimized trajectories are smooth in position, velocity, and acceleration of the MAV. Key for accelerating the optimization process is a good guess for an initial trajectory and its derivatives. We employ a simple

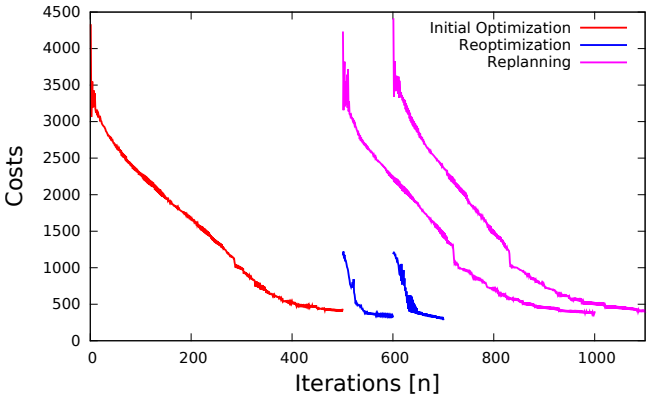

Fig. 10: Frequent reoptimization allows for quick reactions on deviations while following a trajectory. The initial trajectory is planned and optimized for 500 iterations (red). Reoptimizing the old trajectory yields a close-to-optimal new trajectory with fewer iterations than complete replanning.

motion derived from the acceleration capabilities of the MAV, combined with cubic spline interpolation. This reduces the necessary initial optimization effort drastically and allows for fast convergence to a locally optimal and globally feasible trajectory.

Acknowledgment

This work was funded by the German Federal Ministry for Economic Affairs and Energy (BMWi) in the Autonomics for Industry 4.0 project InventAIRy.

References

[1] D. Droeschel, M. Nieuwenhuisen, M. Beul, D. Holz, J. Stückler, and S. Behnke, "Multi-layered mapping and navigation for autonomous micro aerial vehicles," *Journal of Field Robotics*, available online, 2015.

[2] D. Droeschel, J. Stückler, and S. Behnke, "Local multi-resolution surfel grids for MAV motion estimation and 3D mapping," in *Int. Conf. on Intelligent Autonomous Systems (IAS)*, 2014.

[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," *International Journal of Robotics Research*, vol. 32, pp. 1164–1193, 2013.

[4] S. Bouabdallah, P. Murrieri, and R. Siegwart, "Design and control of an indoor micro quadrotor," in *IEEE Int. Conf. on Robotics and Automation (ICRA)*, 2004.

[5] T. Puls, M. Kemper, R. Kuke, and A. Hein, "GPS-based position control and waypoint navigation system for quadrocopters," in *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 2009.

[6] S. Grzonka, G. Grisetti, and W. Burgard, "A fully autonomous indoor quadrotor," *IEEE Trans. on Robotics*, vol. 28, no. 1, pp. 90–100, 2012.

[7] C. Goerzen, Z. Kong, and B. Mettler, "A survey of motion planning algorithms from the perspective of autonomous uav guidance," *Journal of Intelligent & Robotic Systems*, vol. 57, no. 1-4, pp. 65–100, 2010.

[8] J. Israelsen, M. Beall, D. Bareiss, D. Stuart, E. Keeney, and J. van den Berg, "Automatic collision avoidance for manually tele-operated unmanned aerial vehicles," in *IEEE Int. Conf. on Robotics and Automation (ICRA)*, 2014.

[9] 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," *Journal of Field Robotics*, vol. 31, no. 4, pp. 654–675, 2014.

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

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

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

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

[14] K. Ok, S. Ansari, B. Gallagher, W. Sica, F. Dellaert, and M. Stilman, "Path planning with uncertainty: Voronoi uncertainty fields," in *IEEE Int. Conf. on Robotics and Automation (ICRA)*, 2013.

[15] F. Andert, F.-M. Adolf, L. Goormann, and J. S. Dittrich, "Autonomous vision-based helicopter flights through obstacle gates," in *Selected papers from the 2nd International Symposium on UAVs*. Springer, 2010, pp. 259–280.

[16] M. S. Whalley, M. D. Takahashi, J. W. Fletcher, E. Moralez, L. C. R. Ott, L. M. G. Olmstead, J. C. Savage, C. L. Goerzen, G. J. Schulein, H. N. Burns, and B. Conrad, "Autonomous Black Hawk in flight: Obstacle field navigation and landing-site selection on the RASCAL JUH-60A," *Journal of Field Robotics*, vol. 31, no. 4, pp. 591–616, 2014.

[17] E. N. Johnson and J. G. Mooney, "A comparison of automatic nap-of-the-earth guidance strategies for helicopters," *Journal of Field Robotics*, vol. 31, no. 4, pp. 637–653, 2014.

[18] I. A. Şucan and L. E. Kavraki, "Kinodynamic motion planning by interior-exterior cell exploration," in *Algorithmic Foundation of Robotics VIII, Int. Workshop on the Algorithmic Foundations of Robotics (WAFR)*, ser. Springer Tracts in Advanced Robotics, vol. 57. Springer, 2008, pp. 449–464.

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

[20] I. A. Şucan, M. Moll, and L. E. Kavraki, "The Open Motion Planning Library," *IEEE Robotics & Automation Magazine*, vol. 19, no. 4, pp. 72–82, 2012.

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

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

[23] 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 Int. Conf. on Intelligent Robots and Systems (IROS)*, 2012.

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

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

[26] M. Kalakrishnan and K. Anderson, "MoveIt: Propagation distance field," Online available: github.com/ros-planning/moveit_core.

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

[28] D. Holz, M. Nieuwenhuisen, D. Droeschel, M. Schreiber, and S. Behnke, "Towards multimodal omnidirectional obstacle detection for autonomous unmanned aerial vehicles," in *Int. Arch. Photogramm. Remote Sens. Spatial Inf. Sci. (ISPRS)*, vol. XL-1/W2, 2013, pp. 201–206.