Efficient Trajectory Planning for High Speed Flight in Unknown Environments

Markus Ryll[1], John Ware[1], John Carter[1] and Nick Roy[1]

***Abstract*— There has been considerable recent work in motion planning for UAVs to enable aggressive, highly dynamic flight in known environments with motion capture systems. However, these existing planners have not been shown to enable the same kind of flight in unknown, outdoor environments. In this paper we present a receding horizon planning architecture that enables the fast replanning necessary for reactive obstacle avoidance by combining three techniques. First, we show how previous work in computationally efficient, closed-form trajectory generation method can be coupled with spatial partitioning data structures to reason about the geometry of the environment in real-time. Second, we show how to maintain safety margins during fast flight in unknown environments by planning velocities according to obstacle density. Third, our receding-horizon, sampling-based motion planner uses minimum-jerk trajectories and closed-loop tracking to enable smooth, robust, high-speed flight with the low angular rates necessary for accurate visual-inertial navigation. We compare against two state-of-the-art, reactive motion planners in simulation and benchmark solution quality against an offline global planner. Finally, we demonstrate our planner over 80 flights with a combined distance of 22 km of autonomous quadrotor flights in an urban environment at speeds up to 9.4 ms^{-1}.**

I. INTRODUCTION

Although the planning problem for systems with simple dynamics and known maps is well understood, trajectory planning for vehicles with non-trivial dynamics in cluttered and unknown environments is an active field of research. These systems arise in applications such as autonomous navigation of ground and aerial vehicles, warehouse and medical robotics, and mobile manipulators. As of yet, very few approaches have been able to address this problem robustly and in real-time for high-speed quadrotor flight.

In this work, we consider the problem of online planning for a micro aerial vehicle (MAV) in an unknown, cluttered, and GPS-denied environment using on-board sensing and computation. Because on-board sensors have a limited sensing horizon, the planner must be able to react sufficiently quickly to discovered obstacles in order to guarantee safety. We therefore seek to find a smooth, high-speed, collision-free and dynamically feasible 3D trajectory within a timeframe amenable to highly reactive flight.

Online path planning for MAVs is particularly challenging due to the size, weight, and power constraints inherent in such platforms. While optimization-based planners (e.g. [1], [2]) might find optimal solutions (e.g. shortest path, minimum energy [3]), they suffer from high computational costs, uncertain solve times, and frequently require good initialization. These properties make them less suitable for the problem addressed in this work. In contrast, sampling-based planners (e.g. [4]–[6]) have a predictable execution time, but the solution quality is limited by the choice of the sampling measure. Although there are several relevant examples of sampling based approaches (e.g. [4], [7]), these planners seek to minimize computational load by relying solely on instantaneous sensor measurements. These approaches allow for high replan rates but the limited sensing horizon of the depth sensor makes the planner myopic. This can lead to becoming trapped in local minima and poor situational awareness. As shown in [7], the varying obstacle densities present in the type of environments considered here additionally require the planner to adjust its speed to safely navigate through heavy clutter. Difficulties arise when the need for aggressive flight has to be balanced with the requirements of the visual-inertial state estimation system. It has been shown that vision systems perform best under relatively smooth camera motion [8]. Unlike the existing sampling-based motion planners already mentioned, our planner produces a smooth final trajectory to support the visual-inertial navigation system. It has been noted that this allows visual-inertial odometry systems to increase the number of trackable features between camera frames and reduce the discontinuities in position and rotation between state estimator outputs [9]. A further benefit is that smooth trajectories reduce the energy consumption compared to lower order trajectories [10].

This work attempts to combine many of the best qualities of optimization and sampling-based planners by using a hierarchical planning architecture that considers both, the instantaneous and fused history of depth information. Although not the focus of this work, this planner relies on a coarse global planner, discussed in greater detail in Sec. III-B, to provide a local goal along a sparsified optimal path through the mapped environment. Note that the single source, shortest path algorithm used in the global planner allows the local planner to avoid local minima. In order to evaluate the motion

Fig. 1: Quadrotor performing aggressive flight maneuvers autonomously in urban environment. Hardware and sensors include an NUC Skull canyon, an Intel RealSense RGB-D camera and a Flea3 mono camera.

[1]CSAIL, Massachusetts Institute of Technology, Cambridge, USA {ryll,jakeware,jcarter,nickroy}@csail.mit.edu

We thank Brett Lopez and Pete Florence for sharing their planners, which allowed the conducted comparisons. This work was supported by J-C Lede and the DARPA FLA program under Contract No. HR0011-15-C-0110.

primitives with respect to this local goal, we must first fuse the available metric depth information into a 3D occupancy grid and voxel-grid filter the instantaneous depth image. To perform efficient radius queries, we insert both the fused map and voxel-grid filtered depth image into separate K-D trees. Using the quadrotor's desired state sampled along the trajectory $50\,\mathrm{ms}$ in the future ($\approx 30\,\mathrm{ms}$ algorithm completion time + $20\,\mathrm{ms}$ safety margin), we generate a set of motion primitives with different final positions, aiming for a large spatial coverage over the depth camera's field of view. The point $50\,\mathrm{ms}$ in the future, the concatenation point, is the point from which we start tracking the next primitive. We rank the primitives by evaluating them against the K-D trees of both the instantaneous and fused depth information and the endpoints' proximity to the local goal. Finally, we select the lowest cost candidate primitive and join it with the current trajectory at the previously selected concatenation point.

Our approach is capable of replanning at a rate of $30\,\mathrm{Hz}$, and to the best of the authors' knowledge, this is the fastest update rate for a trajectory planner on a UAV which considers a map. The planning algorithm presented here has been demonstrated on a quadrotor (Fig. 1) over $22\,\mathrm{km}$ of flights in unknown, urban environments.

II. DEFINITIONS AND PROBLEM FORMULATION

We now present the problem described in Sec. I formally. Consider $\mathbf{x}_s(t) \in \mathcal{X} \subset \mathbb{R}^{3n}$ to be a 3D dynamical system state with its $(n-1)$ derivatives. Let us further define $\mathcal{X}^{\text{free}} \subset \mathcal{X}$ as kinodynamically feasible regions of the state space capturing not only the free space $\mathcal{P}^{\text{free}}$ but constraints $\mathcal{D}^{\text{free}}$ of the dynamical system, i.e., peak velocity $v_{\max}$ and constraints in higher derivatives. Note that $\mathcal{P}^{\text{free}}$ is bounded in the planning space. Given the dynamical constraints, we can define the feasible state space as $\mathcal{X}^{\text{free}} = \mathcal{P}^{\text{free}} \times \mathcal{D}^{\text{free}}$.

Let us next discuss the dynamical model of a quadrotor. As commonly done we will use a 3D rigid body model to describe the quadrotor. Although more sophisticated models exist which include aerodynamic effects, a rigid body model is a sufficient approximation even for high-speed flights due to the nature of our receding horizon planning approach. Let us define a world frame $\mathcal{F}_W$ and quadrotor bodyframe $\mathcal{F}_R$. $\mathcal{F}_R$ coincides with the center of mass of the rigid body, resulting in three translational degrees of freedom $\mathbf{p} = [p_1\ p_2\ p_3]^\intercal \in \mathbb{R}^3$ and three rotational degrees of freedom. The rotational degrees of freedom, with respect to $\mathcal{F}_R$, can be represented by a rotation matrix $\mathbf{R} \in \mathbb{R}^{3\times 3}$. We define as control inputs to the system the thrust force scalar $f \in \mathbb{R}$ and the three body fixed rotational torques $\boldsymbol{\omega} = [\omega_1\ \omega_2\ \omega_3]^\intercal \in \mathbb{R}^3$. We can fairly assume that high bandwidth controllers are able to track well the attitude commands due to the vehicle's low inertia. Therefore the full state of the quadrotor can be described by 9 variables — the vehicle's position, velocity and orientation. As commonly [11] done, the full dynamics of the quadrotor body can now be expressed as

$$\ddot{\mathbf{p}} = \mathbf{R}\mathbf{e}_3 f + \mathbf{g} \tag{1}$$

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

with $\ddot{\mathbf{p}}$ being the translational acceleration of the quadrotor, $\mathbf{g} = [0\ 0\ -mg]^\intercal$ being the gravitational acceleration vector, $\mathbf{e}_3 = [0\ 0\ 1]^\intercal$ aligns the force scalar f with the propeller orientation and $[]_{[\times]}$ is the skew-symmetric operator.

The differential flatness property of standard multirotor systems allows us to construct trajectories [12] by composing polynomials independently about the three multirotor position axes. Let us consider a polynomial

$$\mathbf{p}_D(t) = \sum_{k=0}^{K} \mathbf{d}_k \frac{t^k}{k!} = \mathbf{d}_K \frac{t^K}{K!} + \cdots + \mathbf{d}_1 t + \mathbf{d}_0 \in \mathbb{R}^3 \tag{3}$$

with $\mathbf{D} = [\mathbf{d}_0, \ldots, \mathbf{d}_K] \in \mathbb{R}^{3\times(K+1)}$. The desired polynomial trajectory and its derivatives can now be found as $\mathbf{x}_D(t) = [\mathbf{p}_D(t)^\intercal, \dot{\mathbf{p}}_D(t)^\intercal, \ldots, \mathbf{p}_D^{(n-1)}(t)^\intercal]$. The desired trajectory $\mathbf{x}_D(t)$ as constructed in (3) is dynamically feasible if and only if the actuation limits for the total thrust $0 \leq f_{\min} \leq f \leq f_{max}$ in (1) and the magnitude of the angular velocity $||\boldsymbol{\omega}|| \leq \omega_{max}$ in (2) are not violated. By concatenating $N \in \mathbb{Z}^+$ polynomials $\mathbf{p}_D(t)$, we can find a trajectory that connects the starting and the goal location. We then find the shortest path from the start to a goal location along the minimum cost path, defined by N concatenated polynomials $\mathbf{p}_D(t)$, while minimizing the jerk $\dddot{\mathbf{p}}_D(t)$ along the trajectories and obeying the kinodynamic and geometric constraints ($\mathbf{x}_D(t) \in \mathcal{X}^{\text{free}}$).

III. TRAJECTORY COMPUTATION AND SELECTION

In the following we will describe the full trajectory planning algorithm in detail and how it solves the optimization problem in Sec. II. The full motion planning algorithm is composed by five steps:

- Compute a set of minimum-jerk motion primitives based on a future state along the robot's current trajectory
- Find a sparsified, global 3D path to the goal location based on the occupancy grid
- Select a local goal on the global path that serves as the receding horizon goal for the motion primitives. The distance of the local goal to the quadrotor depends on the vehicle's speed
- Evaluate the motion primitives with respect to dynamic feasibility, depth image, occupancy grid and distance to the moving local goal and choose the best based on a weighted cost function
- Follow the motion primitive with the minimum cost until a new depth image arrives

All steps of the planning algorithm and their associated update rates are depicted in Fig. 2 and described in detail in the following.

A. Minimum-jerk motion primitive set generation

At every planning step, we generate a spatially and dynamically diverse set of motion primitives in $\mathcal{X}^{\text{free}}$, by sampling over yaw-heading, final altitude, and distance from the vehicle such that we uniformly cover the area with the depth

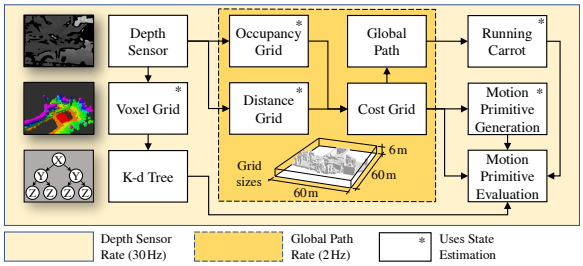

Fig. 2: Planning stack framework. Bright yellow elements are triggered by the depth image rate. The dark yellow elements are part of the global path planning and run at 2 Hz.

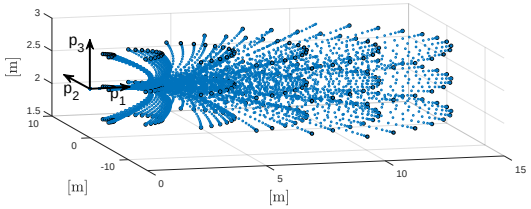

Fig. 3: Example of valid motion primitives: The minimum-jerk primitives have been selected with 15 different heading angles, six different final velocities and three different final altitudes. The final position of every trajectory is highlighted in black. All displayed motion primitives are dynamically feasible and collision-free.

camera's field of view (see Fig. 3)[1]. Sampling over speeds naturally occurs by using a constant time across all candidate trajectories. In order to account for the computation time associated with trajectory sampling and generation, we select the motion primitives' initial state as that of the current trajectory $50\,\mathrm{ms}$ in the future. This point in the future is the concatenation point from which we will start tracking the next motion primitive. Because we only consider primitives with a fixed final velocity and acceleration, the full set of motion primitives is defined by the set of final positions $\mathcal{P}_f$:

$$\begin{aligned} \mathcal{P}_f &= \{\mathbf{p}_{f_1} \dots \mathbf{p}_{f_m}\} \text{ with } m = n \cdot o \cdot p \\ \mathbf{p}_{f_i} &= \mathbf{R}_z(\psi_k) l_j \mathbf{e}_1 + h_l \mathbf{e}_3 \ , \\ j \in \{1 \dots n\} \ , \ k \in \{1 \dots o\} \ , \ l &\in \{1 \dots p\} \end{aligned} \tag{4}$$

with $\mathcal{L} = \{l_0, \dots, l_n\}$ changing the final position and respectively the peak velocity, $\psi = \{\psi_0, \dots, \psi_o\}$ changing the yaw heading (represented as an Euler angle), $\mathcal{H} = \{h_0, \dots, h_p\}$ changing the altitude and $\mathbf{e}_n$ being the unit vector along the n-th axis. An illustrative example of sets of minimum-jerk trajectories along a followed trajectory is displayed in Fig. 3. For the simulated and real experiments in Sec. IV we used a set of 270 motion primitives, covering a large spatial and dynamical space.

The generation of each motion primitive follows the computationally efficient, closed-form solution presented in [13], [14], based on fifth order polynomials. A motion primitive is defined by an initial and final state (position, velocity and acceleration) and is thrice differentiable. Next, the dynamical feasibility of the trajectory is tested following the approach in [13]. Infeasible trajectories are not in the set of dynamically feasible trajectories $\mathcal{D}^{\text{free}}$.

[1]Yaw-Euler angles are used to sample the motion primitives, while the controller internally uses a rotation matrix representation.

B. Planning grids and moving local goal

In order to decouple the local obstacle avoidance from the global guidance, the planner presented here relies on a global planner, running in a separate process, to provide it with a local goal. Note that this local goal acts as the navigation goal for our receding horizon trajectory planner. To maintain a sufficiently high spatial resolution within the flight computer's computational capabilities, the global planner maintains two body-centered, 3D voxel grids with dimensions of $60\,\mathrm{m} \times 60\,\mathrm{m} \times 6\,\mathrm{m}$ at a resolution of $0.5\,\mathrm{m}$ in all dimensions. The first grid is a probabilistic occupancy grid [15] containing a fused history of depth sensor measurements. The second grid uses the incremental distance transform presented in [16] with a short truncation distance to keep track of the proximity to the nearest obstacle for every voxel. The global planner runs a breadth-first search [17] on a weighted sum of the occupancy grid, distance grid, and a user-specified target altitude, to extract a minimum-cost path to the voxel nearest to the global goal at $2\,\mathrm{Hz}$. This path is then sparsified and used to generate a local goal for the trajectory planner with a lookahead distance that is proportional to the quadrotor's speed.

Finally we evaluate the motion primitives and select the one with the lowest cost. We split every primitive in n control points corresponding to the resolution of the occupancy grid. We first perform operations that are fast to compute and directly eliminate primitive candidates. Therefore, we check if the control points of the primitives are in collision with the occupancy grid or the depth image. Trajectories that are in collision are excluded. Second, we add a cost to every primitive depending on the number of control points outside or in unknown space of the occupancy grid. Third, we add a cost, linearly increasing with the distance d_{goal} between the local goal and the primitive's final position. Fourth, for every control point we add a cost linearly increasing with the distance d_{alt} to the desired altitude. Last, we add a cost depending on the distance of obstacles to the motion primitive. Which requires us to find all obstacles in a fixed distance d to the control points. To do this efficiently we convert the occupancy grid into a K-D tree and an approximate nearest neighbor search [18], [19] is used to get the all neighbors along the n control points within the distance d. In average we query the K-D tree 1300 times per motion primitive set generation with an average query time of $0.02\,\mathrm{ms}$. The average time for all queries, including the K-D tree building time, is $2.4\,\mathrm{ms}$ while a linear search requires $6.9\,\mathrm{ms}$. The approximate nearest neighbor search is therefore a factor of 3 faster than a conventional linear search. The total cost c assigned to every primitive is

$$c = \sum_{i=1}^{n} (c_{\text{outside}} + c_{\text{unknown}} + k_1 d_{\text{alt}} + c_{\text{prox}}) + k_2 d_{\text{goal}}$$

$$c_{\text{prox}} = \sum_{j=1}^{m} k_3 \frac{v}{d_{\text{obs}}^2} \begin{cases} k_3 = 1, & \text{if obstacle in distance } d \text{ to control point} \\ k_3 = 0, & \text{otherwise} \end{cases}$$

with c_i being the individual costs, k_j being individual gains, m the number of occupied cells in the occupancy grid and

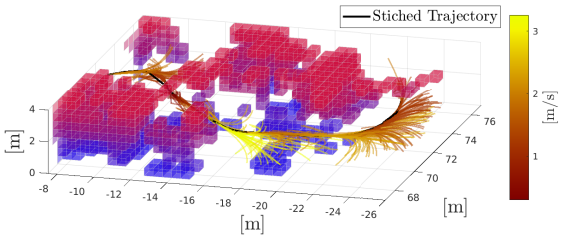

Fig. 4: Example of the concatenated trajectory in a cluttered environment: Occupancy grid is visualized by blocks and the concatenated trajectory by the black line. The colored lines are the single motion primitives, where only the initial part is tracked. The color coding shows the peak velocity of the primitive.

v the initial velocity of the primitive. Finally, the trajectory with the lowest cost is selected.

C. Receding planning horizon

To achieve a reactive planner we compute the full set of motion primitives at every new depth image (e.g., at 30 Hz). Only the first 33 ms of the selected motion primitive is tracked. An illustrative example of the single primitives and the concatenated trajectory is shown in Fig. 4. As we use the replan state (position, velocity and acceleration) of the vehicle to compute the new set of motion primitives, the concatenation point of the trajectories is at least continuous to the third order of the polynomial.

IV. RESULTS

We present simulated and real experiments. To point out single, unique features of the planner, we will present simulated results. In the real flight test section we will present data from a single, representative flight followed by a thorough analysis of a large set of conducted flights.

A. Simulation results

We present simulation results for three different experiments. First, we compare the smoothness of the trajectories generated to those of two other state-of-the-art approaches to this problem. Second, we compare solution quality to that of an optimal offline planner. Third, we show results testing the reactivity and the limits of the maximum velocity. All simulated experiments have been conducted running the actual flight stack, including low-level control, perception and state estimation. The dynamics and the IMU are simulated using DRAKE [20]. The perception is simulated by using a high fidelity Unity[2] environment.

1) Trajectory smoothness: By using continuous minimum-jerk trajectories and fast re-planning, we achieve a higher smoothness of the trajectory compared to other planners. In the following we compare our planner with the planners by Lopez et al. [4], a minimum-time, state-space sampled planner for aggressive maneuvers and Florence et al. [7], a control-space-sampled motion primitive planner. Both planners are selected for comparison as they represent highly reactive planners with frequent re-planning. We follow the same trajectory with all planners in simulation,

[2]Game engine by Unity Technologies: https://unity3d.com

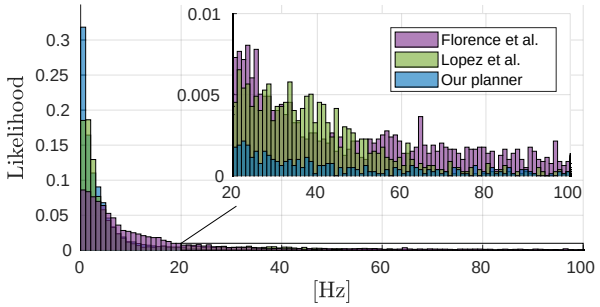

Fig. 5: Comparison of the angular rate of our planner versus Lopez et al. [4]. and Florence et al. [7]. Data are represented in a likelihood histogram over the roll and pitch frequencies (Fast Fourier Transform) of an identical mission. It is clearly visible that our planner utilizes much lower frequencies, being very beneficial for any VIO system. For better visibility the order of the three planners is swapped in the zoomed plot.

with an average velocity of $4\,\mathrm{m\,s^{-1}}$, while dodging several obstacles. The results are presented in Fig. 5.

Our planner's angular rate frequencies are significantly lower. While 90 % of the frequencies are below 10 Hz these are only 77 % at Lopez's planner and at 59 % at Florence's planner.

2) Path quality: To assess the ability of our planner to find short trajectories, found trajectory we compared it against RRT* with 250, 1000 and 5000 samples. Although path length is not a term of the objective function, it is still operational relevant. A Poisson forest like environment was used with 100 randomly distributed and oriented cylindrical obstacles (see Fig. 6 - top). The Poisson forest has a dimension of 100 m by 60 m. The starting and stopping point were each 20 m away from the long side of the forest (distance between start and stop point is 100 m). The forest itself has a density of 12 % and therefore includes many dead ends. Dead ends are challenging for planners as they represent local minima.

In total we compared our planner in 50 randomly generated maps against the three RRT* planners. We would like to point out that the RRT* paths are not dynamically feasible for multirotor systems but require a smoothing of the path. However, RRT* serves as a very good comparison of the found trajectory length. RRT* with 250 samples did not succeed in finding a feasible path in 34.0 % of the trials while RRT* with 1000 and 5000 samples succeeded in all trials. Our planner's path is on average 18.2 % shorter as RRT* 1000 and 21.6 % shorter as RRT* 250 in the cases where RRT* found a solution (see Fig. 6 - bottom) but 1.8 % longer than RRT* with 5000 samples. RRT* with 5000 samples is already close to the optimal solution and therefore serves as a good baseline for the length evaluation of our planner. The run time to compute the RRT* paths exceeded our planner by several magnitudes, therefore RRT* would not be feasible for real time applications. Due to the lack of publicly available implementations we could not test our planner against other planners discussed in Sec. V.

3) Planner reactivity: We evaluated the reactivity of the planner and compared it with a lower re-planning frequency. The vehicle was commanded to fly in a straight line while

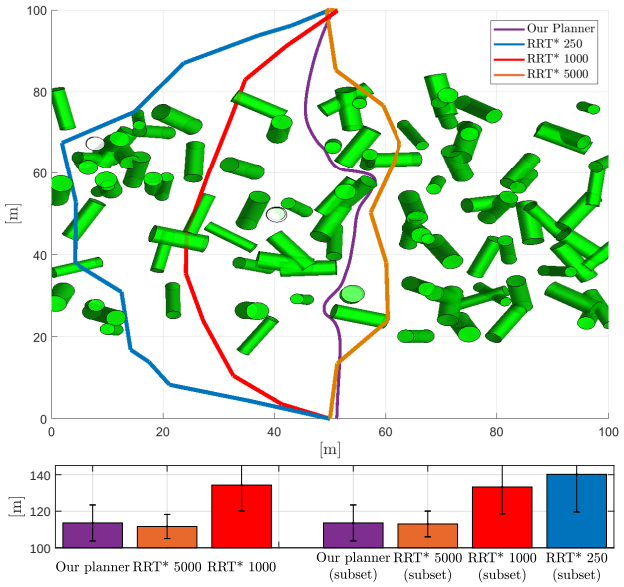

Fig. 6: Path search through Poisson forest like environment. Top: Example of randomly generated Poisson forest with trajectory of our planner (lilac), an RRT* path, based on 250 samples (blue), an RRT* path, based on 1000 samples (red) and an RRT* path, based on 5000 samples (orange). Bottom: Mean trajectory length and standard deviation of 50 randomly generated maps. RRT* with 250 samples failed in 39 % of the maps to find a path. We therefore present the subset of maps where RRT* 250 successfully found a trajectory in the right three plots.

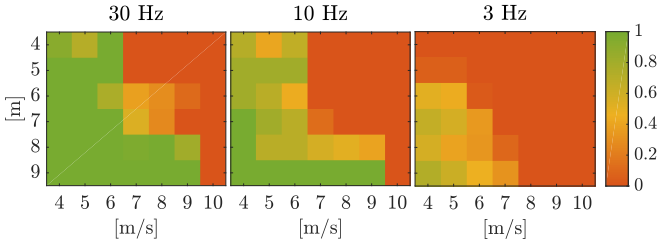

Fig. 7: Likelihood from zero to one to pass an obstacle, depending on initial velocity (x-axes) and distance to the obstacle (y-axis) and re-planning frequency (left 30 Hz, center 10 Hz, right 3 Hz). The obstacle is a vertical cylinder with 1 m diameter.

a vertical cylindrical obstacle (diameter 1 m) instantaneously appeared in front of the vehicle. Different initial velocities (4-9 $\mathrm{m\,s^{-1}}$) and distances to the obstacle were simulated (4-10 m). The vehicle is supposed to avoid the obstacle. In total ≈400 trials were conducted for all three re-planning frequency conditions. The results are presented in Fig. 7. Intuitively, avoiding the obstacle became harder with increasing velocity or decreasing distance to the obstacle. It is interesting, however, that the likelihood of a crash increases with a decreasing control frequency, underlining the importance of a high re-planning rate. At the nominal control frequency of 30 Hz all obstacles appearing in 9 m distance and at a vehicle velocity $9\,\mathrm{m\,s^{-1}}$ could be avoided. With a control frequency of 3 Hz only 28 % of the obstacles could be avoided.

4) Dynamic velocity planning: To demonstrate the benefits of the dynamic velocity planning of our planner, we compared our planner against itself with fixed velocities. Therefore the quadrotor has been commanded to traverse the

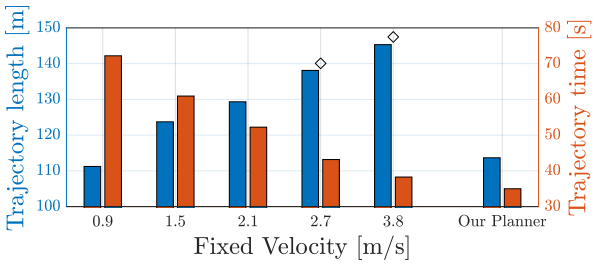

Fig. 8: Trajectory length and trajectory time duration for different, fixed velocities in comparison to our planner. Velocities marked with a ◇ resulted partially in collisions with obstacles.

Poisson forest of Sec: IV-A.2. We selected motion primitives with only a fixed length l in (4). Since the trajectory length is fixed this results in a constant velocity over the full trajectory — the planner is not able to do dynamic velocity planning. We conducted the experiment individually for the trajectory lengths $\mathcal{L} = \{3, 5, 7, 9, 12\}[m]$. The trajectory length and durations generated by the planner using these fixed velocities are presented in Fig. 8.

With an increasing fixed velocity, the time to traverse the Poisson forest decreases (red bars in Fig. 8) while the tracked trajectory length increases (blue bars in Fig. 8). Starting from velocities higher than $3\,\mathrm{m\,s^{-1}}$ an increasing number of trajectories are in collision with obstacles, showing how important it is for a planner to be able to reduce flight velocity. Our planner with dynamic velocities found trajectories through the Poisson forest with a length almost as short as in the case for $1\,\mathrm{m\,s^{-1}}$ but shorter in time than in the case for $3.8\,\mathrm{m\,s^{-1}}$.

B. Experimental flight tests

The presented work is a result of the MIT-Draper FLA[3]-project. The mechanical hardware includes a quadrotor Flamewheel DJI frame, for visual-inertial state estimation an ADIS 16448 IMU and a monocular Point Grey Flea3 camera (640x512 pixel, 30 Hz), a D435 Intel RealSense depth camera (RGB image and depth image: 640x480 pixel, 30 Hz) and a quad core Intel NUC Skull Canyon i7 with an i7-6770HQ CPU as the flight computer. The total weight of the platform is approximately 3.2 kg allowing a flight time of 6 min. The flights were conducted at the Guardian Centers of Georgia (USA) in an urban-like environment.

1) Discussion of a Single Exemplary Flight: The fully autonomous mission included six relative defined waypoints in an unknown urban environment and required the vehicle to return to the start location (Fig. 9-left). The flight covered a distance of 720 m and lasted 3 min 39 s. Although the roads were relatively wide, the waypoints forced the vehicle to navigate around several buildings, trees, cars, and smaller obstacles such as streetlights (see way points 3-5 in Fig. 9-top). A particularly challenging segment of the flight occurred between waypoints 3 and 4, where the vehicle had to enter and traverse a 2.8 m wide and 9.4 m long pathway between two buildings (Fig. 9-right). This tight section of the environment resulted in an average velocity

[3]DARPA Fast Lightweight Autonomy program: https://www.darpa.mil/program/fast-lightweight-autonomy

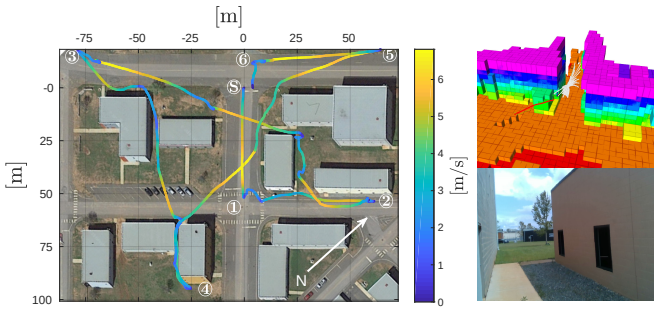

Fig. 9: Flight tests at Guardian Centers. Left: Exemplary flight trajectory. The 'S' marks the starting point and numbers present the given waypoints. The trajectory color encodes the speed. As desired, the velocity increases in wide, open spaces, while decreases in narrow passages and turns. Right top: Visualization of the Occupancy Grid at entrance of narrow passage, next to waypoint 4. Right bottom: Camera view of the narrow passage.

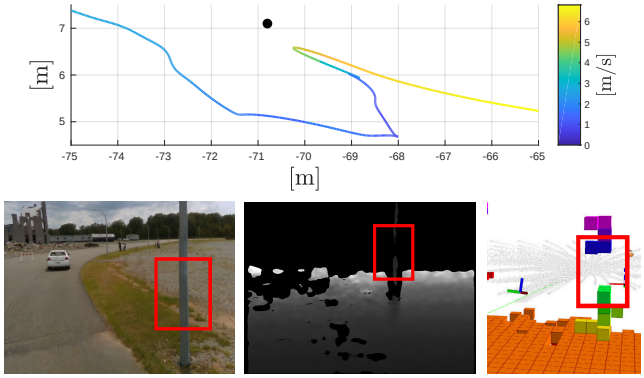

Fig. 10: Aggressive maneuver resulting from a false negative hole in the lamppost. Top figure: Trajectory while avoiding a lamppost obstacle. The vehicle conducts an emergency breaking maneuver and backs off of the obstacle. The line color indicates the norm of the velocity. The black dot indicates the approximated position of the obstacle lamppost. Bottom figures: (a) Camera image while approaching the lamppost. (b) Depth image, the pole is split into an upper and a lower part. (c) Path leading to collision.

of 2.4 m/s, but the vehicle achieved a peak velocity of 8 m/s in more open areas. The average velocity for the whole flight was $3.2\,\mathrm{m\,s^{-1}}$. Finally, the vehicle performed a particularly aggressive maneuver shortly before waypoint 3 in order to dodge a light pole. While much of the light pole was clearly visible in the depth image (see Fig. 10-(a)), the depth camera reported a false negative at the flight altitude. This caused the planned trajectories to pass through the obstacle (see Fig. 10-(b),(c)) while the vehicle was traveling at $6\,\mathrm{m\,s^{-1}}$. Luckily, the depth camera detected the obstacle in time for the planner to perform an emergency stop and back away from the pole with a peak acceleration of $6\,\mathrm{m\,s^{-2}}$.

2) Discussion on full, 22 km flight data: For this work we conducted 72 flights with a total length of $22\,\mathrm{km}$ and a total flight duration of $105.5\,\mathrm{min}$. During the flights we had two vehicle crashes, both caused by loading the wrong calibration files at the ground station, unrelated to the trajectory planning. We achieved a peak velocity of $8.9\,\mathrm{m\,s^{-1}}$. The longest covered distance in a single mission was $743.8\,\mathrm{m}$ while the longest mission lasted $6.0\,\mathrm{min}$. The velocity distribution of all flights is depicted in Fig. 11.

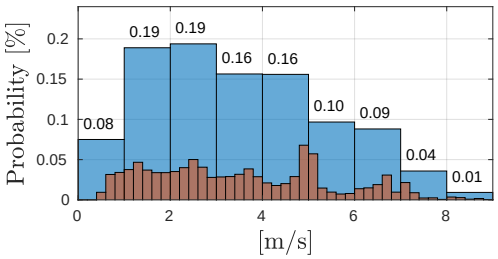

Fig. 11: Velocity distribution of 72 flights. Velocities below $0.5\,\mathrm{m\,s^{-1}}$ are omitted as they are dominated by the starting and landing procedure.

V. RELATED WORK

Although there exist many planning approaches, capable of flight speeds below $3\,\mathrm{m\,s^{-1}}$, relatively few address high-speed quadrotor flight. We will now present a short summary of works on high velocity flight, in unknown environments.

Fraundorfer et al. [21] and Shen et al. [22] were the first presenting a fully vision-based mapping system for high-speed MAVs. Daftry et al. [23] performed robust monocular flights in highly cluttered environments. Richter et al. [24] demonstrated highly aggressive flights in indoor environments. Aggressive indoor maneuvers through narrow gaps have been shown by Falanga et al. [25]. Lopez et al.'s [4] Triple Integrator Planner computes very fast state space sampled motion primitives resulting in a very reactive planner optimized for obstacle avoidance during fast flight. Mohta et al. [26] present a whole quadrotor framework for high-speed, aerial navigation including an extensive section on the onboard VIO system. The approach reached velocities up to $18\,\mathrm{m\,s^{-1}}$, currently the fastest autonomous flight of a multi-rotor using vision-based state estimation in clutter-free environments. This work has been extended in [27]. High speed velocity flight and obstacle avoidance with a fixed wing has been presented by Barry et al. [28].

VI. CONCLUSIONS AND FUTURE WORK

We presented a receding horizon trajectory planner for fast flight in unknown and cluttered environments. Our approach is capable of reasoning about the geometry of the environment in real-time while maintaining safety margins during fast, smooth, and robust flight. We compared against two state-of-the-art, reactive motion planners in simulation and benchmarked solution quality against an offline global planner. We demonstrated our planner's capabilities over 80 flights with a combined distance of $22\,\mathrm{km}$ of autonomous quadrotor flights in an urban environment at speeds up to $9.4\,\mathrm{m\,s^{-1}}$. As future work, we will seek to further reduce the vehicle's planning horizon and exploit all available information in the local occupancy grid by increasing the search depth to incorporate sequences of multiple minimum-jerk trajectories in a single planning step. Furthermore we plan to make the planner publicly available.

REFERENCES

[1] M. Watterson and V. Kumar, "Safe receding horizon control for aggressive mav flight with limited range sensing," in *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Sept 2015, pp. 3235–3240.

[2] F. Gao, Y. L. William Wu, and S. Shen, "Online safe trajectory generation for quadrotors using fast marching method and bernstein basis polynomial," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*, May 2018.

[3] F. Morbidi, D. Bicego, M. Ryll, and A. Franchi, "Energy-efficient trajectory generation for a hexarotor with dual-tilting propellers," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, May 2018, p. TBD.

[4] B. T. Lopez and J. P. How, "Aggressive 3-d collision avoidance for high-speed navigation," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*, May 2017, pp. 5759–5765.

[5] S. Liu, N. Atanasov, K. Mohta, and V. Kumar, "Search-based motion planning for quadrotors using linear quadratic minimum time control," in *Intelligent Robots and Systems (IROS), 2017 IEEE/RSJ International Conference on*. IEEE, 2017, pp. 2872–2879.

[6] S. Liu, K. Mohta, N. Atanasov, and V. Kumar, "Search-based motion planning for aggressive flight in se(3)," *IEEE Robotics and Automation Letters*, vol. 3, no. 3, pp. 2439–2446, July 2018.

[7] P. Florence, J. Carter, and R. Tedrake, "Integrated perception and control at high speed: Evaluating collision avoidance maneuvers without maps," in *WAFR 2016*, 2016.

[8] T. J. Steiner, R. D. Truax, and K. Frey, "A vision-aided inertial navigation system for agile high-speed flight in unmapped environments: Distribution statement a: Approved for public release, distribution unlimited," in *2017 IEEE Aerospace Conference*, March 2017, pp. 1–10.

[9] T. Qin, P. Li, and S. Shen, "Vins-mono: A robust and versatile monocular visual-inertial state estimator," *IEEE Transactions on Robotics*, vol. 34, no. 4, pp. 1004–1020, 2018.

[10] N. Kreciglowa, K. Karydis, and V. Kumar, "Energy efficiency of trajectory generation methods for stop-and-go aerial robot navigation," in *2017 International Conference on Unmanned Aircraft Systems (ICUAS)*, June 2017, pp. 656–662.

[11] T. Lee, M. Leoky, and N. H. McClamroch, "Geometric tracking control of a quadrotor uav on se (3)," in *Decision and Control (CDC), 2010 49th IEEE Conference on*. IEEE, 2010, pp. 5420–5425.

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

[13] M. W. Mueller, M. Hehn, and R. D'Andrea, "A computationally efficient motion primitive for quadrocopter trajectory generation," *IEEE Transactions on Robotics*, vol. 31, no. 6, pp. 1294–1310, Dec 2015.

[14] M. Hehn and R. D'Andrea, "Quadrocopter trajectory generation and control," *IFAC Proceedings Volumes*, vol. 44, no. 1, pp. 1485–1491, 2011.

[15] A. Elfes, "Using occupancy grids for mobile robot perception and navigation," *Computer*, no. 6, pp. 46–57, 1989.

[16] S. Scherer, J. Rehder, S. Achar, H. Cover, A. Chambers, S. Nuske, and S. Singh, "River mapping from a flying robot: state estimation, river detection, and obstacle mapping," *Autonomous Robots*, vol. 33, no. 1, pp. 189–214, Aug 2012.

[17] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to algorithms*. MIT press, 2009.

[18] J. L. Blanco and P. K. Rai, "nanoflann: a C++ header-only fork of FLANN, a library for nearest neighbor (NN) wih kd-trees," https://github.com/jlblancoc/nanoflann, 2014.

[19] M. Muja and D. G. Lowe, "Scalable nearest neighbor algorithms for high dimensional data," *IEEE Transactions on Pattern Analysis & Machine Intelligence*, no. 11, pp. 2227–2240, 2014.

[20] R. Tedrake and the Drake Development Team, "Drake: A planning, control, and analysis toolbox for nonlinear dynamical systems," 2016.

[21] F. Fraundorfer, L. Heng, D. Honegger, G. H. Lee, L. Meier, P. Tanskanen, and M. Pollefeys, "Vision-based autonomous mapping and exploration using a quadrotor mav," in *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*. IEEE, 2012, pp. 4557–4564.

[22] S. Shen, Y. Mulgaonkar, N. Michael, and V. Kumar, "Vision-based state estimation and trajectory control towards high-speed flight with a quadrotor." in *Robotics: Science and Systems*, vol. 1. Citeseer, 2013.

[23] S. Daftry, S. Zeng, A. Khan, D. Dey, N. Melik-Barkhudarov, J. A. Bagnell, and M. Hebert, "Robust monocular flight in cluttered outdoor environments," *arXiv preprint arXiv:1604.04779*, 2016.

[24] C. Richter, A. Bry, and N. Roy, "Polynomial trajectory planning for aggressive quadrotor flight in dense indoor environments," in *Robotics Research*. Springer, 2016, pp. 649–666.

[25] D. Falanga, E. Mueggler, M. Faessler, and D. Scaramuzza, "Aggressive quadrotor flight through narrow gaps with onboard sensing and computing using active vision," in *Robotics and Automation (ICRA), 2017 IEEE International Conference on*. IEEE, 2017, pp. 5774–5781.

[26] K. Mohta, K. Sun, S. Liu, M. Watterson, B. Pfrommer, J. Svacha, Y. Mulgaonkar, C. J. Taylor, and V. Kumar, "Experiments in fast, autonomous, gps-denied quadrotor flight," *arXiv preprint arXiv:1806.07053*, 2018.

[27] K. Mohta, M. Watterson, Y. Mulgaonkar, S. Liu, C. Qu, A. Makineni, K. Saulnier, K. Sun, A. Zhu, J. Delmerico, K. Karydis, N. Atanasov, G. Loianno, D. Scaramuzza, K. Daniilidis, C. J. Taylor, and V. Kumar, "Fast, autonomous flight in gps-denied and cluttered environments," *Journal of Field Robotics*, vol. 35, no. 1, pp. 101–120.

[28] A. J. Barry, P. R. Florence, and R. Tedrake, "High-speed autonomous obstacle avoidance with pushbroom stereo," *Journal of Field Robotics*, vol. 35, no. 1, pp. 52–68, 2018.