Safe Local Exploration for Replanning in Cluttered Unknown Environments for Microaerial Vehicles

Helen Oleynikova, Zachary Taylor, Roland Siegwart, and Juan Nieto

***Abstract*—In order to enable microaerial vehicles (MAVs) to assist in complex, unknown, unstructured environments, they must be able to navigate with guaranteed safety, even when faced with a cluttered environment they have no prior knowledge of. While trajectory-optimization-based local planners have been shown to perform well in these cases, prior work either does not address how to deal with local minima in the optimization problem or solves it by using an optimistic global planner. We present a conservative trajectory-optimization-based local planner, coupled with a local exploration strategy that selects intermediate goals. We perform extensive simulations to show that this system performs better than the standard approach of using an optimistic global planner and also outperforms doing a single exploration step when the local planner is stuck. The method is validated through experiments in a variety of highly cluttered environments including a dense forest. These experiments show the complete system running in real time fully onboard an MAV, mapping and replanning at 4 Hz.**

***Index Terms*—Motion and path planning, aerial systems, perception and autonomy, visual-based navigation.**

I. Introduction

MICRO-AERIAL Vehicles (MAVs) have the potential to perform many mapping and inspection missions for search and rescue and other humanitarian operations, where it is dangerous or impractical for humans to go. Planning is a key part of any autonomous system, and online local replanning allows for fast reactions to newly observed or dynamic parts of the environment. And while local replanning has also been recently addressed in literature, most work is shown on very low-density scenes, and makes optimistic assumptions about the environment (for example, that unknown space can be treated as free before observing it) [1], [2].

However, in more cluttered, unknown environments, these assumptions may lead to poor planning results. Executing these plans can also be dangerous, both for the MAV and nearby people. For example, assuming unknown space is free in forest spaces can lead to planning directly upwards into the tree canopy, which can occur as obstacles directly above an MAV are often outside the field of view of its sensors. Alternatively if a highly conservative local planner is employed, many cluttered environment will result in the system finding no feasible paths to the goal. In this work, we present a system that combines a conservative local planner with a local exploration strategy to navigate a cluttered, unknown environment such as the forest in Fig. 1.

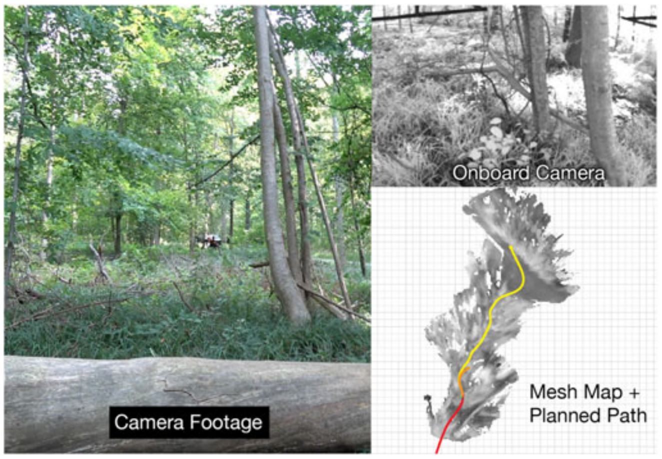

Fig. 1. Experimental results from a flight through a dense forest, with a video camera footage on the left, on-board view from one of the stereo cameras on the upper right, and a representation of the final mesh map on the bottom right. The final flown path is shown in yellow, the current pose of the MAV in the photos is shown as colored axes, and the planned path at the time of the photo is shown in orange.

Different local optimization methods for avoidance have been recently covered in literature [3]–[5]. However, most do not explicitly address the problem of getting stuck in local minima. This poses a special problem in unexplored or partially unexplored environments, where only locally-optimal or reactive planners will frequently fail to find a path. Other approaches use an optimistic global planner (one that considers unknown space as free) to overcome the problem of occasionally getting stuck. While this works well in low-density environments, our work aims to show that this strategy (using an optimistic RRT* [6] for goal selection) is not effective for highly-cluttered, partially unexplored environments.

Instead, we bring in concepts from the exploration literature to the area of local replanning. We compare optimistic global

Manuscript received September 10, 2017; accepted January 15, 2018. Date of publication January 31, 2018; date of current version February 22, 2018. This letter was recommended for publication by Associate Editor S. Chakravorty and Editor N. Amato upon evaluation of the reviewers' comments. This work was supported in part by armasuisse, in part by the National Center of Competence in Research Robotics, in part by the Swiss State Secretariat for Education, Research and Innovation under Contract 15.0029, and in part by the European Community's Seventh Framework Program under Grant 608849 (EuRoC) and Grant 644227 (Flourish). *(Corresponding author: Helen Oleynikova.)*

The authors are with the Autonomous Systems Lab, ETH Zürich, 8092 Zürich, Switzerland (e-mail: helen.oleynikova@mavt.ethz.ch; ztaylor@ethz.ch; rsiegwart@ethz.ch; jnieto@ethz.ch).

This letter has supplemental downloadable multimedia material available at http://ieeexplore.ieee.org, provided by the authors. The Supplementary Materials contain a video of the experiments and a short summary of the approach. This material is 40 MB in size.

Digital Object Identifier 10.1109/LRA.2018.2800109

planning to performing an exploration step from the exploration-gain-based "next-best-view" planner (NBVP) when the trajectory optimization planner fails to find a feasible solution [7]. We then propose our own local exploration method, which tightly couples the local planning algorithm with a strategy that selects an intermediate goal. The method maximizes both coming closer to the final goal and potential exploration gain, increasing the chances of finding a feasible path.

To solve the problem of map representations, our method also uses an incrementally-built, dynamically-growing Euclidean Signed Distance Field (ESDF) to compute collision costs and gradients. The ESDF is built from a Truncated Signed Distance Field (TSDF) [8], and allows us to plan in initially unknown environments with no prior knowledge of upper bounds on map size, and does not require pre-computing the object distances in batch.

We compare different parameters for our underlying local optimization method, which is an extension of our previous work [3], when the map is initially unknown, and then compare the success rates of various intermediate goal-finding strategies in highly cluttered environments. We then demonstrate our complete system running in real-time on-board an Asctec Firefly MAV and navigating without any prior map knowledge through both an office environment and a dense forest.

The contributions of this work are as follows:

- Extension of optimization problem for continuous-time polynomial trajectory optimization.
- A system, including mapping and planning, which conservatively handles unknown space and is able to grow the map over time.
- An active *local* exploration strategy for overcoming local minima even in unknown environments by finding intermediate goal points.
- Simulation benchmarks and real-world experiments in various cluttered environments.

II. Related Work

While a large number of methods exist for local avoidance, we will address methods in 3 categories. The first is purely reactive methods, which do not build a map of the environment but instead plan directly in the current sensor data. While these methods are very fast and computationally efficient, they do not work well in cluttered environments where avoidance maneuvers may be non-trivial, and suffer heavily from falling into local minima. The second class is map-based local avoidance methods, which use various techniques to compute feasible and locally-optimal paths through local maps built from sensor data or *a priori* known global maps. The last class of work we will examine here does not focus on obstacle avoidance, but instead on maximizing exploration coverage of unknown environments. While planning collision-free paths is also a requirement for any exploration strategy, the focus is on minimizing unknown space in the final map. We will draw inspiration from some of these methods to overcome the shortcomings of using optimization-based local planners alone.

A. Reactive Avoidance

Reactive methods focus on reacting to incoming sensor data as quickly as possible, and so act directly on obstacles in the current sensor field of view without building persistent maps.

For instance, our previous reactive work shows a method to directly convert incoming disparity maps from stereo into object segmentations, and then uses wall-following algorithm to avoid them [9]. Florence *et al.* directly integrates the nearest obstacle from a disparity map into a controller that is an open-loop library of motion primitives [10]. Only inexact, local state estimation is required for this approach, and they demonstrate it in both extensive simulation and real-world experiments. Lopez and How build a kD tree of the current sensor view pointcloud, and then perform aggressive reactive avoidance from a library of fixed-velocity but variable angle motion primitives, generated from a triple-integrator model of MAV dynamics [11].

While all three methods are shown avoiding obstacles directly in front of the MAV without prior map knowledge, they are only demonstrated on much lower obstacle densities than discussed in this letter, and suffer from not being able to avoid obstacles that are not directly in the current sensor field of view.

B. Map-Based Replanning

In contrast, most replanning methods focus on navigating in a map rather than directly on sensor data.

Richter *et al.* presented dynamics-aware path planning for MAVs as solving an unconstrained QP through a visibility graph generated by an RRT [12], which remains a popular method for global planning [13], but is debatably too slow to replan in real-time. Our previous work [3] combines unconstrained polynomial spline optimization with gradient-based minimization of collision costs from CHOMP [14], but is prone to local minima. Usenko *et al.* utilize a similar concept, but use a B-spline representation instead, and use a circular buffer-based Octomap to overcome the issue of needing a fixed map size [4]. Dong *et al.* also use the same general problem structure as CHOMP, but represents trajectories as samples drawn from a Gaussian Process (GP) and optimize the trajectory using factor graphs and probabilistic inference [5]. While all these methods are able to avoid obstacles and replan in real time, none offer convincing ways to overcome the problem of getting stuck in a local minima and being unable to find a feasible solution.

Pivtoraiko *et al.* use graph search with motion primitives to replan online [2]. However, they use an optimistic local planner: unknown space is considered traversible, and while this helps escape local minima, it is fundamentally unsafe. Chen *et al.* plan online by building a sparse graph by inflating unoccupied corridors within an Octomap, then optimize an unconstrained QP to get a polynomial path [1]. However, they only use 2D sensing and treat unknown space as free, again leading to potentially unsafe paths in very cluttered environments.

C. Exploration

The goal of exploration literature is not only to stay safe and avoid collisions, but to maximize the amount of information about the environment. There are many different approaches, such as greedily tracking the closest unexplored frontier [15] or simulating gas-like particles throughout the environment to find the sparsest area of dispersion to explore [16].

Rather than tracking frontiers, some methods instead aim to maximize information gain. Charrow *et al.* optimize this gain over a state lattice with motion primitives as connecting edges, and then improve the plan with trajectory optimization [17]. Bircher *et al.* instead build an RRT tree in the unexplored space, and execute a straight-line plan to the first vertex of the most promising branch of the tree, maximizing the number of unknown voxels falling into the sensor frustum [7]. Papachristos *et al.* extend Bircher's method by also optimizing the intermediate paths to maximize localization quality [18]. Similarly, Davis *et al.* optimize paths between next-best views to maximize coverage by introducing a coverage term to their iLQG formulation [19].

Our work combines the fast online replanning capabilities of trajectory optimization-based planning with the idea of maximizing exploration gain in a future sensor field of view. This combination allows us to overcome the tendency of local planners to get stuck with local minima, while intelligently using our model of the system to find feasible solutions.

III. Problem Description

We aim to solve the problem of an MAV attempting to reach a goal in a previously unexplored (and completely unknown) environment. The core focus being on very obstacle-dense and cluttered environments, with forest flight as a particular example. The MAV has at least one 3D imaging sensor, either RGB-D or stereo, with a finite resolution and a fixed horizontal and vertical FOV, mounted in a fixed position. We assume that the MAV is building a map of the environment from this sensor as it navigates (Section V). We design a conservative local planner, which treats unknown space as occupied and inaccessible (Section IV). The core problem we want to address is how to design a complementary goal-finding algorithm for when the local planner gets 'stuck' in a local minimum (Section VI). All parts of the method should be fast enough to run online and in real-time entirely on-board the MAV.

IV. Local Trajectory Optimization

Our local trajectory optimization method is an extension of our previous work [3]. We represent an MAV trajectory as a high-degree polynomial spline as in Richter *et al.* [12], and put soft constraints (expressed in the segment time allocation) on the maximum velocity and acceleration along the trajectory, which Mellinger *et al.* show makes the trajectory physically feasible for a simplified dynamics model [20].

The actual optimization minimizes a compound cost, consisting of minimizing a derivative of position such as jerk or snap as in [12] and [20], combined with the collision gradient cost from Ratliff *et al.* [14].

We will consider a polynomial trajectory in K dimensions, with S segments, and each segment of order N. Each segment has K dimensions, each of which is described by an Nth order polynomial:

$$f_k(t) = a_0 + a_1 t + a_2 t^2 + a_3 t^3 \dots a_N t^N \tag{1}$$

with the polynomial coefficients:

$$\mathbf{p}_k = \begin{bmatrix} a_0 \; a_1 \; a_2 \; \dots \; a_N \end{bmatrix}^\top. \tag{2}$$

In order to avoid numerical issues with high orders of t, we instead optimize over the end-derivatives of segments within the spline [12], sorted into fixed derivatives $\mathbf{d}_F$ (such as end-constraints) and free derivatives $\mathbf{d}_P$ (such as intermediate spline connections):

$$\mathbf{p} = \mathbf{A}^{-1}\mathbf{M}\begin{bmatrix} \mathbf{d}_F \\ \mathbf{d}_P \end{bmatrix}. \tag{3}$$

Where $\mathbf{A}$ is a mapping matrix from polynomial coefficients to end-derivatives, and $\mathbf{M}$ is a reordering matrix to separate $\mathbf{d}_F$ and $\mathbf{d}_P$.

The final form of the optimization problem is:

$$\mathbf{d}_P^* = \underset{\mathbf{d}_P}{\operatorname{argmin}} \; w_d J_d + w_c J_c + w_g J_g \tag{4}$$

Where the derivative cost, J_d, aims to minimize a certain derivative (often jerk or snap) of the position [20], with $\mathbf{R}$ as the augmented cost matrix.

$$J_d = \mathbf{d}_F^\top \mathbf{R}_{FF}\mathbf{d}_F + \mathbf{d}_F^\top \mathbf{R}_{FP}\mathbf{d}_P + \mathbf{d}_P^\top \mathbf{R}_{PF}\mathbf{d}_F + \mathbf{d}_P^\top \mathbf{R}_{PP}\mathbf{d}_P \tag{5}$$

The collision cost, J_c, is an approximation of the line integral of costs along the path, where $c(\mathbf{x})$ is the collision cost from the map, $\mathbf{f}(t)$ is the position along the trajectory at time t, and $\mathbf{v}(t)$ is the velocity at time t:

$$J_c = \sum_{t=0}^{t_m} c(\mathbf{f}(t)) \left\| \mathbf{v}(t) \right\| \Delta t \tag{6}$$

We use 3 segments and optimize jerk, as was found to be the best settings in our previous work [3].

We extend our previous work by using a soft cost for the goal, J_g, similarly to [4], and the local goal finding below.

$$J_g = \left\| \mathbf{f}(t_{\text{end}}) - \mathbf{g} \right\| \tag{7}$$

where

$$f_k(t_{\text{end}}) = \mathbf{T}_{\text{end}}\mathbf{A}^{-1}\mathbf{M}\begin{bmatrix} \mathbf{d}_{Fk} \\ \mathbf{d}_{Pk} \end{bmatrix} \tag{8}$$

This allows the optimization to slightly adjust the goal point to allow better trajectories, or find feasible trajectories at all. An analysis of the effect of this term on the success rate is offered in Section VII.

In general, even with the soft cost term, the initial state of the optimization problem should have the end point be free or almost free of collisions. In our system, we set a fixed planning horizon r_p, which is the maximum distance from the current state that the planner is allowed to go to. However, projecting a global goal $\mathbf{g}_g$ onto the sphere of this radius often leads to occluded end points.

In Section VII, we compare two different strategies for moving this end-goal to be a feasible end point for the spline: straight-line goal finding, which backtracks along the line from the projection of $\mathbf{g}_g$ to the start point of the trajectory, $\mathbf{x}_s$, until the first unoccupied point along this line. The second method is gradient-based in the map: from the projection of $\mathbf{g}_g$ onto the sphere, we evaluate the gradient of the collision cost map and follow the gradient down until a free-space location is found. If the approach becomes stuck in a local minimum of the gradient, we evaluate the straight-line strategy for one step.

Finally, these trajectories are only planned on $\mathbb{R}^3$ and derivatives. To map these trajectories to the full pose of the MAV on SE^3, pitch and roll are defined by the acceleration in x and y directions, while yaw γ remains free. We use velocity-tracking yaw to increase the chances of the MAV seeing new or dynamic obstacles before collision:

$$\gamma(t) = \arctan\left(\frac{v_y(t)}{v_x(t)}\right) \tag{9}$$

V. Map Representation and Unknown Space

As the optimization method in Section IV requires not only distances to the nearest obstacles but also the gradients of these distances, we require a map representation that can be efficiently queried for this information. While our original work [3] used a fixed-size Euclidean Signed Distance Field (ESDF) built from an octomap representation, we more recently presented a way to build ESDFs from Truncated Signed Distance Fields (TSDFs) efficiently. This allows the system to incrementally build maps of arbitrary size from sensor data in real time. This system, called *voxblox*,[1] is used as the map representation for the proposed planner [8].

The map consists of both the original TSDF, built from sensor data, which contains projective signed distances to surfaces within a very small truncation distance to the object and free space information, and the ESDF which contains Euclidean distances to obstacles in a much larger radius. The details of how to build both representations incrementally is addressed in [8].

To implement the desired property of treating unknown space as occupied, we modify the ESDF with data from the current state of the robot. One critical issue with treating unknown space as occupied is that the starting position of the robot will never be observed and will always be treated as occupied. For this reason, we change the ESDF values of unknown voxels in a small clearing radius r_c around the initial pose of the MAV to free. r_c should ideally be only slightly larger than the collision checking radius of the robot.

We also take a large radius r_o, which should be greater than or equal to the maximum planning radius, and set all unknown voxels in this radius to occupied. Marking unknown as occupied is essential to conservative local planners, as allowing free entry into unknown space leads to behaviors such as slamming into the ceiling when presented with obstacles in front.

VI. Intermediate Goal Selection

In addition to the mapping and local planning methods presented above, we need an active exploration strategy to overcome the shortcomings of local trajectory optimization methods in very cluttered, partially unknown environments. A typical solution to this problem is to use an optimistic global planner, which assumes unknown space is free, to select a new set of waypoints to track [21].

In the results section (Section VII), we quantitatively compare five core methods of selecting new waypoint locations. The first method is naive random waypoint selection. When the local optimization fails, we select a new 3D waypoint position at random within a sphere of the starting position of the trajectory. The planner then attempts to track this waypoint, until it is either reached or another infeasible solution is encountered. Then the new waypoint is set to the original goal point. This strategy (one random, one back to original goal) continues until either the original goal point is reached or the maximum number of replans is exceeded.

The second strategy is an optimistic (unknown = free) RRT* [6] visibility graph. This aims to best simulate the global planners used in other approaches, such as [4] and [13], [21]. Since as Section V describes, we set a large radius of unknown space to occupied in the ESDF, we instead use the raw TSDF as the obstacle map and treat unknown voxels as unoccupied. We then generate a sparse visibility graph toward the final goal, and track the first waypoint in the graph. If the first waypoint is reached, then we keep iterating through the graph until the goal point. If at any time, the local planner is stuck again, we generate a new RRT* plan.

We also consider the opposite strategy: a conservative or pessimistic RRT*, which assumes unknown space is occupied. Since the underlying local planner is also conservative, if it is unable to find a solution, it is likely that no solution to the goal exists through free space. Therefore, we build the RRT graph and select the node in the tree that has the closest Euclidean distance to the goal point, and then track the first vertex in the branch of the tree that the closest node belongs to.

The next strategy we consider is directly from the exploration literature, the "next-best view" planner (NBVP) from Bircher *et al.* [7]. Their approach consists of building a rapidly-exploring random tree (RRT) with a small number of nodes in position and yaw space, and simulating the expected view fustrum of the camera sensor. The approach then selects the first node to execute in the branch that leads to the highest information gain in terms of unknown voxels that would be observed. We implement this approach for comparison; however, since there is no goal-tracking component to this exploration strategy, we use the same scheme as with the random waypoint selection: one exploration waypoint, followed by trying to reach the goal, followed by another exploration waypoint.

The final strategy is our exploration strategy, combining aspects of both the exploration strategy above and goal-tracking and sensor field of view awareness, described in detail below.

A. Proposed Method

Our method uses a similar methodology to NBVP, where the potential exploration gain of future points is evaluated by projecting the camera frustum into the voxel grid. However, we adapt the method to (i) better suit the purpose of increasing the

[1]github.com/ethz-asl/voxblox

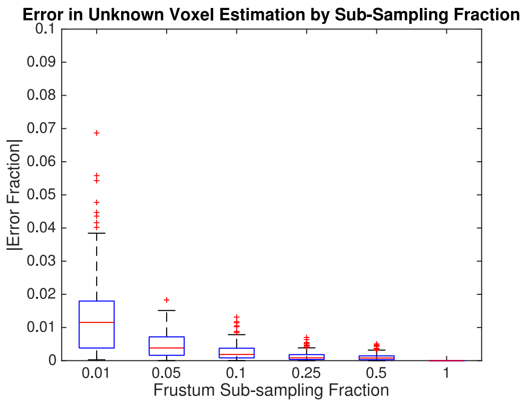

Fig. 2. Error in estimation of unknown voxels in the sensor frustum (as a proxy for exploration gain), by subsampling fraction (a subsampling fraction of $0.01 = 1\%$ of the samples are taken). As can be seen, a sampling of 5% of the samples yields only a maximum 2% error in the unknown voxel estimation but could lead to up to a $20\times$ speedup in lookup operations.

chances of the robot making it to the goal, and (ii), to function online, in real-time in a high-rate loop. The core differences are that we do not build an RRT graph, we subsample within the view frustum, we do not do raycasting to find occlusions, and we introduce a goal-seeking reward in addition to the exploration gain.

Our method works as follows: first, we draw the global goal $\mathbf{g}$ with some probability $P_g \in (0, 1)$. Otherwise, we proceed to generate N random points, $\mathbf{x}_n$, in the *unoccupied* space of the TSDF, within a maximum radius r of the start point of the trajectory $\mathbf{x}_s$. Note, importantly, that we use the original TSDF rather than the ESDF to select these points and evaluate the frustum, as the ESDF sets nearby unknown space to occupied for planning safety purposes.

We select a yaw γ for each point by finding the angle of the vector from the trajectory start $\mathbf{x}_s$ to the sampled point $\mathbf{x}_n$, to approximate the real velocity-facing yaw. For each of these points, we evaluate the exploration gain of the camera frustum at that point by counting the number of unknown voxels in the TSDF. The exploration gain function $l(\mathbf{x}, \gamma)$ can be expressed as:

$$l(\mathbf{x}, \gamma) = \#\{v | v \in \text{frustum}\,(\mathbf{x}, \gamma) \cap v \in \text{unknown}\,(v)\} \quad (10)$$

In order to run in real-time, we approximate the actual exploration gain by subsampling the frustum by a certain factor, and checking only every sth voxel. We evaluate the effect of this approximation in Fig. 2 in simulation, which shows that sampling only 5% of the samples usually leads to an estimation error of less than 1%, and in practice runs 3 times faster than evaluating the full frustum.

Additionally, for each point we also evaluate the distance to the global goal, normalized by the maximum distance to goal d_g (to allow consistent weighting across different settings and goal distances). This normalized distance is converted to a reward, giving the total reward function R for each point $\mathbf{x}_n$ as:

$$d_g = \|\mathbf{g} - \mathbf{x}_s\| + r \quad (11)$$

$$R(\mathbf{x}_n, \gamma, \mathbf{g}) = w_e l(\mathbf{x}, \gamma) + w_g \frac{d_g - \|\mathbf{g} - \mathbf{x}_n\|}{d_g} \quad (12)$$

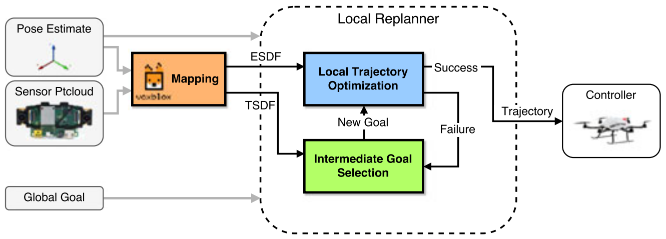

Fig. 3. System diagram of the mapping and planning subsystems. The ESDF is used by the trajectory optimizer to compute collision costs, and the TSDF is used by the intermediate goal finding (local exploration algorithm) to evaluate exploration gain. If the trajectory optimization succeeds, the trajectory is sent to the controller; otherwise, we attempt to find an alternative intermediate goal.

The point with the highest reward is chosen as the next intermediate goal.

A diagram showing the complete system (including mapping) is shown in Fig. 3.

VII. Simulation Experiments

This section will evaluate different aspects of our system in a simulation environment where the ground truth map is known. We evaluate the effect of parameters on success rate of local trajectory optimization, compare the intermediate goal finding methods presented in Section VI, and the effect of subsampling the camera view frustum for exploration gain evaluation.

These simulations are made with the *voxblox*, which allows generating ground-truth ESDFs for environments made of primitive shapes (in this case, cylinders to simulate trees in a forest), and also allows simulating sensor measurements by raycasting into the map. The maps are 15×10 meters, and have an obstacle region of 10×10, to ensure that the start and end poses are always free. Cylinders of radii between 0.1 and 0.5 m and various heights are placed randomly within the space. The objects per square meter metric maps to approximately to percentage of the volume occupied, $\pm 5\%$ (for instance, 0.4 objects/m^2 is 35–45% occupied volume).

For the purposes of these experiments, we assume our MAV can track the polynomial trajectories perfectly, which [20] shows is possible as long as we respect maximum velocity and acceleration bounds while planning. We add a new viewpoint into the incrementally-built map and then replan once a second of simulation time. The incremental planning methods have a maximum planning horizon of 3 meters.

The first results are for starting with an a completely empty map, and inserting new viewpoints along the path at 1 Hz, with a sample solutions from no incremental goal-finding shown in Fig. 4(a) and the quantitative results in Fig. 5. As can be seen, straight-line goal finding and purely local optimization are able to solve only a very small percentage of the test cases. Using gradient-based goal finding significantly increases the performance, and soft goals further increase success rate, especially at lower densities. However, the success rates overall are still unacceptably low.

To overcome these issues, we benchmark the intermediate goal finding methods, described in Section VI, on the same simulation cases. Example qualitative results are shown for all

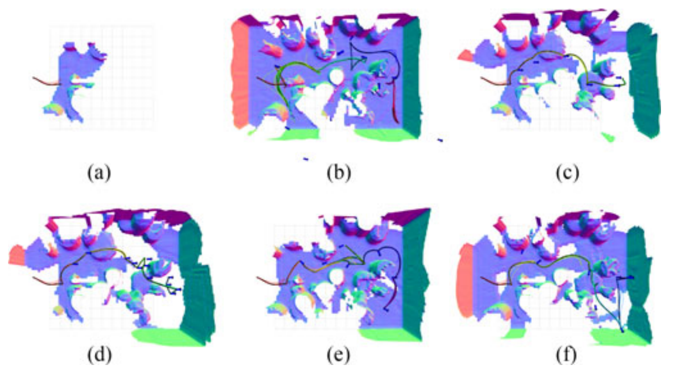

Fig. 4. Comparison of methods in a small simulation case (15 × 10 m) with 0.3 objects/m^2 obstacle density. The black line shows the final path, the colored lines show intermediate paths, and dark blue arrows show the intermediate goals selected by the algorithm. Only our method and NBVP were successfully able to solve the case; both RRT* methods were unable to see the final location as free as they do not consider sensor field-of-view in the planning, and the random goal selection had too few replans. All methods ran for up to 120 replans. (a) No incremental goal. (b) Random goals. (c) RRT* (Opt.). (d) RRT* (Cons.). (e) NBVP. (f) Our method.

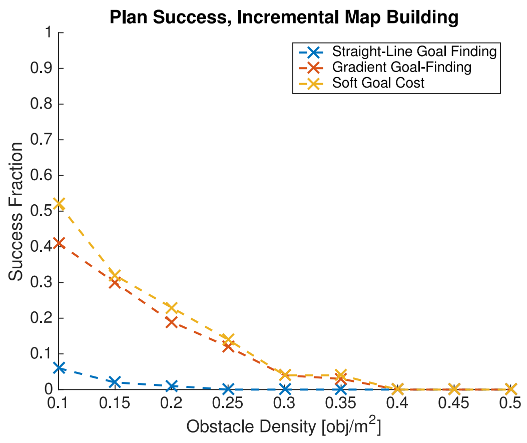

Fig. 5. A comparison of the planner success without any intermediate goal-finding strategy, building the map incrementally. As can be seen, gradient-based goal-finding significantly increases success chance over line-based goal finding, and soft goal-cost further increases performance. There are 100 trials per density, with 60 replans (60 s at 1-Hz replanning rate).

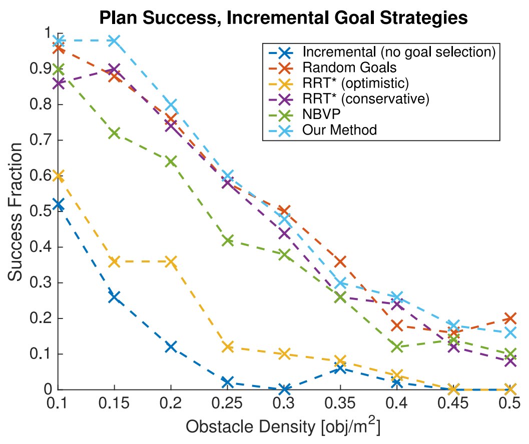

Fig. 6. A comparison of the success rates for different incremental goal-finding strategies. Note that our method, NBVP, and conservative RRT* are able to solve more test cases than optimistic RRT*, which is commonly used as a global planner. There are 50 trials per density, and a maximum of 120 replans per trial.

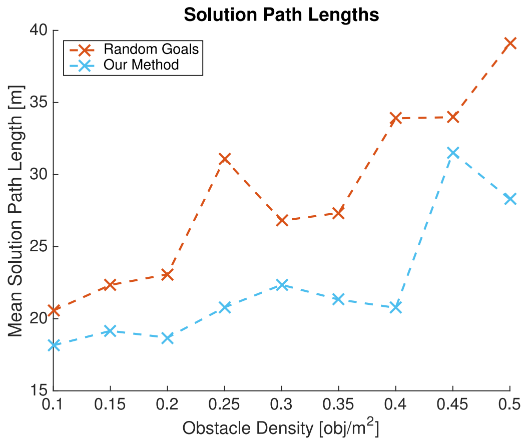

Fig. 7. Path length comparison between random goal selection and our proposed method. The path lengths are only evaluated for trials where both planners succeeded, to allow a fair comparison. Note that our method always finds a solution in a significantly shorter path length, as it exploits current knowledge of the environment.

methods in Fig. 4. The simulations show the differences between the methods: random goals fails to find the goal within the allocated time as the intermediate goals are too undirected, and the two RRT*-based methods fail since they do not consider the field-of-view of the sensor and are therefore never able to observe the goal point as clear.

Fig. 6 shows the quantitative results: as can be seen, all goal-finding methods outperform the naive optimization-only method. The optimistic RRT* performs the worst, as it tends to select the same infeasible path over and over again as unknown space is marked as traversable for this method. NBVP performs somewhat better, as it uses the sensor model to maximize exploring the small area. Conservative RRT* performs comparatively well, as it is simply tracking the closest free point to the goal, but has no knowledge of the sensor model.

Finally, our method performs on par with random goal selection in terms of success rate. However, our method is able to consistently produce much shorter path lengths: Fig. 7 shows the mean path lengths for simulation cases that *both* random goal finding and our method were able to solve. Our method produces paths up to 35% shorter.

The final experiment is a more realistic test of a long forest traversal. We generate a 50 meter × 50 meter randomized map with 0.1 and 0.2 obstacles/m^2, and set the MAV to explore from one corner to the other. The results from 0.2 density are shown in Fig. 8, where our method and optimistic RRT* were the only two to successfully make it to the goal. Simulation results over 20 maps at different densities are shown in Fig. 9, and the timings of different aspects of our method from this simulation are shown in Table I.

VIII. Real-World Experiments

To evaluate our system in a real-world scenario, we performed multiple experiments in two different test environments: a cluttered office space and a dense forest with a variable ground

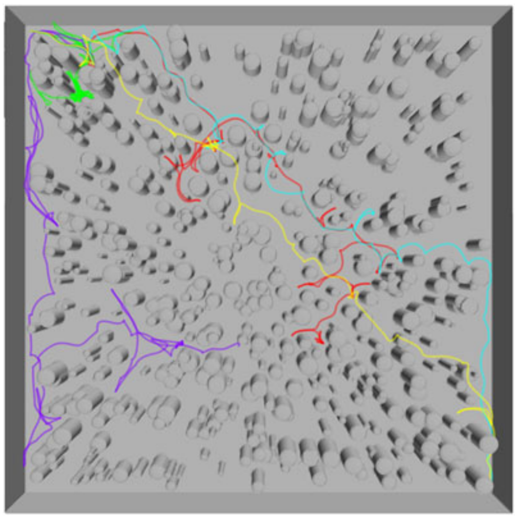

Fig. 8. A 50 m × 50 m randomized "forest" environment, with a density of 0.2 objects/m^2. The lines compare five planners: no goal selection (dark blue), random goal finding (red), optimistic RRT* (yellow), conservative RRT* (purple), NBVP (green), and our method (teal). All planners start in the upper-left corner and try to reach the lower-right, but only our planner and optimistic RRT* are able to successfully find a solution in 500 replan cycles. The planning is in 3-D, so some plans go over an obstacle.

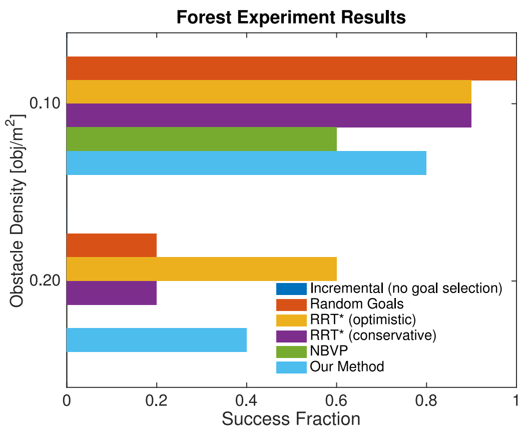

Fig. 9. Quantitative results of success rate from the long forest simulation, limited to 500 replan cycles. While the RRT-based methods can offer good performance in this situation they also sample orders of magnitude more points than our method, and require much more time to select an intermediate goal.

TABLE I
TIMINGS FOR A SINGLE ITERATION EACH PART OF THE METHOD, AGGREGATED FROM THE BENCHMARK IN FIG. 8

Step	**Time** [ms]
Mapping	
TSDF Insert	27.0
ESDF Update	14.5
Local Replanning	
Trajectory Optimization	19.3
Intermediate Goal Selection	5.9

Note that intermediate goal selection will only run if trajectory optimization fails, not every planning iteration.

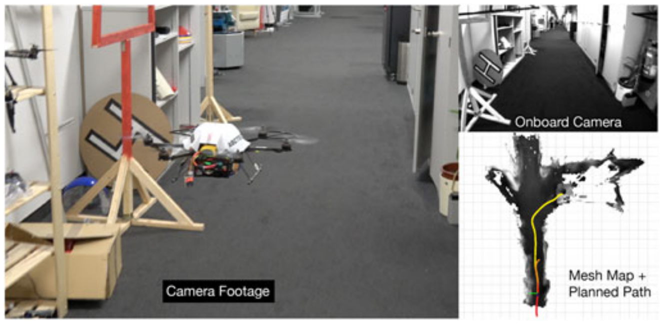

Fig. 10. Experimental results from the office navigation experiment, with final map and intermediate paths shown on the lower right.

height. The results of all described experiments are available at https://youtu.be/rAJwD2kr7c0.

All of the experiments start with a completely unknown map, use visual-inertial odometry from the forward-facing (with a 12° downward pitch) stereo camera, update the map from stereo and replan at 4 Hz, and run everything entirely on the 2.4 GHz i7 dual-core CPU on-board the robot. We use rovio for state estimation [22], a non-linear MPC for position control [23], and the Asctec on-board attitude controller. The average flight velocity was 1.0 m/s.

In the office space environment, the MAV is able to navigate from a starting position in a hallway, around a corner, and to a point above an office table, shown in Fig. 10. During the path, it successfully avoids an ajar cabinet door (which blows open during the flight), along with many obstacles on either side of the hallway. The MAV was only able to reach near the intended goal, as it is unable to successfully determine whether the airspace above the tables is clear or not: the tables are gray and textureless, and the white projector screen behind them is also textureless, leading to a lack of stereo matches and therefore unknown space in the map. While eventually the robot would have explored enough of the surroundings to clear this space, the pilot intervened when it was near the intended target. This demonstrates the conservative and safe nature of our planner.

Our second experimental validation took place in a forest environment, where we performed four different experiments. In the first trial, we were successfully able to avoid a single large tree between the start point and goal. Second, we did two experiments where the MAV was commanded to go a large distance in its current facing direction, where the robot successfully avoided tree branches along its way and navigated largely along a hiking trail for up to 45.0 meters. In the shorter experiment, the MAV was able to reach its goal. In the longer one, it was unable to reach the final goal as the slope of the ground was too high and the tilted-down camera did not allow it to perceive enough open space to safely raise its flying height above the ground.

The final forest experiment tested navigation in very cluttered, obstacle-dense environments. The MAV was commanded to fly in a very densely-forested area between two trails, containing many small trees, branches, uneven terrain, and other obstacles. A still image of the video, along with the corresponding robots-eye view and the final executed path are shown in Fig. 1. The

MAV was able to complete a path of 34.7 meters, successfully avoiding obstacles along the way, and finishing at the waypoint above the trail on the other side of the wooded region.

IX. Conclusion

This letter presented a complete system for local obstacle avoidance, consisting of an underlying trajectory optimization method, which uses an Euclidean Signed Distance Field (ESDF) built by *voxblox* to get collision costs and gradients, coupled with an exploration-inspired intermediate goal finding strategy to escape local minima in the optimization. We showed that our combined method outperforms the common strategy of coupling an optimistic global planner with a conservative local planner. In the case of high obstacle densities, our exploration-based method is able to find solutions to more planning problems. We also outperform the next-best view exploration method for intermediate goal, as we are able to incorporate information about the global goal and reduce the runtime of the exploration gain evaluation.

Our approach focuses on solving the case of very cluttered environments in previously unknown maps, and maximizing the chances of finding the goal while building the map. To demonstrate the performance of our method in real-world scenarios, we were able to successfully navigate through an office and through multiple forest environments while performing all processing in real-time on-board an MAV.

References

[1] J. Chen, T. Liu, and S. Shen, “Online generation of collision-free trajectories for quadrotor flight in unknown cluttered environments,” in *Proc. Int. Conf. Robot. Autom.*, 2016, pp. 1476–1483.

[2] M. Pivtoraiko, D. Mellinger, and V. Kumar, “Incremental micro-UAV motion replanning for exploring unknown environments,” in *Proc. IEEE Int. Conf. Robot. Autom.*, 2013, pp. 2452–2458.

[3] H. Oleynikova, M. Burri, Z. Taylor, J. Nieto, R. Siegwart, and E. Galceran, “Continuous-time trajectory optimization for online UAV replanning,” in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2016, pp. 5332–5339.

[4] V. Usenko, L. von Stumberg, A. Pangercic, and D. Cremers, “Real-time trajectory replanning for MAVs using uniform B-splines and 3D circular buffer,” in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2017, pp. 215–222.

[5] J. Dong, M. Mukadam, F. Dellaert, and B. Boots, “Motion planning as probabilistic inference using Gaussian processes and factor graphs,” in *Proc. Robot.: Sci. Syst. Conf.*, Jun. 2016.

[6] S. Karaman and E. Frazzoli, “Sampling-based algorithms for optimal motion planning,” *Int. J. Robot. Res.*, vol. 30, no. 7, pp. 846–894, 2011.

[7] A. Bircher, M. Kamel, K. Alexis, H. Oleynikova, and R. Siegwart, “Receding horizon “next-best-view” planner for 3D exploration,” in *Proc. IEEE Int. Conf. Robot. Autom.*, 2016, pp. 1462–1468.

[8] H. Oleynikova, Z. Taylor, M. Fehr, R. Siegwart, and J. Nieto, “Voxblox: Incremental 3D Euclidean signed distance fields for on-board MAV planning,” in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2017, pp. 1366–1373.

[9] H. Oleynikova, D. Honegger, and M. Pollefeys, “Reactive avoidance using embedded stereo vision for MAV flight,” in *Proc. IEEE Int. Conf. Robot. Autom.*, 2015, pp. 50–56.

[10] P. Florence, J. Carter, and R. Tedrake, “Integrated perception and control at high speed: Evaluating collision avoidance maneuvers without maps,” in *Proc. Workshop Algorithmic Found. Robot.*, 2016.

[11] B. T. Lopez and J. P. How, “Aggressive 3-D collision avoidance for high-speed navigation,” in *Proc. IEEE Int. Conf. Robot. and Autom.*, 2017, pp. 5759–5765.

[12] C. Richter, A. Bry, and N. Roy, “Polynomial trajectory planning for aggressive quadrotor flight in dense indoor environments,” in *Proc. Int. Symp. Robot. Res.*, 2013, pp. 649–666.

[13] M. Burri, H. Oleynikova, M. W. Achtelik, and R. Siegwart, “Real-time visual-inertial mapping, re-localization and planning onboard MAVs in unknown environments,” in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, Sep. 2015, pp. 1872–1878.

[14] N. Ratliff, M. Zucker, J. A. Bagnell, and S. Srinivasa, “CHOMP: Gradient optimization techniques for efficient motion planning,” in *Proc. IEEE Int. Conf. Robot. Autom.*, 2009, pp. 489–494.

[15] L. Heng *et al.*, “Autonomous visual mapping and exploration with a micro aerial vehicle,” *J. Field Robot.*, vol. 31, no. 4, pp. 654–675, 2014.

[16] S. Shen, N. Michael, and V. Kumar, “Autonomous indoor 3D exploration with a micro-aerial vehicle,” in *Proc. IEEE Int. Conf. Robot. Autom.*, 2012, pp. 9–15.

[17] B. Charrow *et al.*, “Information-theoretic planning with trajectory optimization for dense 3D mapping,” in *Proc. Robot.: Sci. Syst. Conf.*, 2015.

[18] C. Papachristos, S. Khattak, and K. Alexis, “Uncertainty-aware receding horizon exploration and mapping using aerial robots,” in *Proc. IEEE Int. Conf. Robot. Autom.*, 2017, pp. 4568–4575.

[19] B. Davis, I. Karamouzas, and S. J. Guy, “C-OPT: Coverage-aware trajectory optimization under uncertainty,” *IEEE Robot. Autom. Lett.*, vol. 1, no. 2, pp. 1020–1027, Jul. 2016.

[20] D. Mellinger and V. Kumar, “Minimum snap trajectory generation and control for quadrotors,” in *Proc. IEEE Int. Conf. Robot. Autom.*, 2011, pp. 2520–2525.

[21] M. Nieuwenhuisen and S. Behnke, “Layered mission and path planning for MAV navigation with partial environment knowledge,” in *Proc. Int. Conf. Intell. Auton. Syst.*, 2016, pp. 307–319.

[22] M. Bloesch, S. Omari, M. Hutter, and R. Siegwart, “Robust visual inertial odometry using a direct EKF-based approach,” in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2015, pp. 298–304.

[23] M. Kamel, M. Burri, and R. Siegwart, “Linear vs nonlinear MPC for trajectory tracking applied to rotary wing micro aerial vehicles,” arXiv preprint arXiv:1611.09240, 2016.