Rapid Exploration with Multi-Rotors: A Frontier Selection Method for High Speed Flight

Titus Cieslewski, Elia Kaufmann and Davide Scaramuzza

***Abstract*— Exploring and mapping previously unknown environments while avoiding collisions with obstacles is a fundamental task for autonomous robots. In scenarios where this needs to be done rapidly, multi-rotors are a good choice for the task, as they can cover ground at potentially very high velocities. Flying at high velocities, however, implies the ability to rapidly plan trajectories and to react to new information quickly. In this paper, we propose an extension to classical frontier-based exploration that facilitates exploration at high speeds. The extension consists of a reactive mode in which the multi-rotor rapidly selects a goal frontier from its field of view. The goal frontier is selected in a way that minimizes the change in velocity necessary to reach it. While this approach can increase the total path length, it significantly reduces the exploration time, since the multi-rotor can fly at consistently higher speeds.**

Multimedia Material

A video attachment to this work is available at `https://youtu.be/54s6gGZLpJo`.

I. Introduction

Exploring and mapping previously unknown environments is a fundamental task for autonomous robots. Given an environment with free (traversable) and occupied (untraversable) space, the task is to detect free space within a target area. This can be used to map previously unseen environments or to search for objects or people, such as in search and rescue scenarios. The task can be further specified depending on further requirements. In search and rescue, for instance, it is important to find survivors rapidly. Thus, an objective would be to cover the area as quickly as possible. A related objective would be to expend as little energy as possible, for example when mapping with an energy-constrained robot.

Another objective would be to minimize the uncertainty of the map. Many exploration algorithms assume that pose estimation and free space detection are good enough that effects of uncertainty can be ignored, or reduced to the uncertainty of depth sensors only. This assumption, however, cannot be made generally, since real systems do exhibit uncertainty. Exploration algorithms that take uncertainty into account are usually more complex and less generally applicable than algorithms that make the simplifying assumption, since they need to consider specific aspects of the underlying mapping algorithm. In our work, we make this assumption and focus on the case where exploration is performed with a single multi-rotor robot. Most related work in the same setting designs the exploration algorithm in a way that minimizes the distance traveled. At the same time, it is often assumed that the multi-rotor flies at low velocities, which relaxes design constraints when it comes to execution time and trajectory generation. However, a multi-rotor expends energy on hovering and thus its limited energy supply is best used when flying at velocities that are not near-hover.

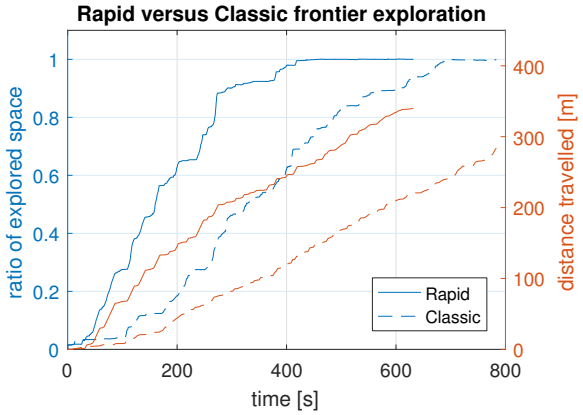

Fig. 1. We propose an algorithm for exploration with multi-rotors at high speeds. While our approach sometimes ends up traveling a larger distance, it is able to complete the exploration task faster, as it is capable of flying at higher speeds, as can be seen in this data of a single run. Note that a lot of time is spent at almost 100% coverage, as the robot needs to travel some distance to complete the final portions of undiscovered space.

In this paper, we propose an exploration algorithm that is designed to fly at high velocities as much as possible. To that end, we introduce a reactive mode, which, instead of planning trajectories generates instantaneous velocity commands based on currently observed frontiers. This control loop is able to run at high frequency, allowing high velocity flight and rapid incorporation of new information. Frontiers that drop out of the current field of view are added to a data structure with global frontiers and we fall back to classical frontier-based exploration as soon as no frontiers are left in the field of view. We evaluate our approach, compare it to previous approaches and show that while it can result in larger distances traveled, the exploration time is lower than with other approaches, in particular at higher velocities.

II. Related work

As previously stated, exploration is the task of detecting free space within a given area. This is achieved with a robot that is capable of detecting free space, typically with a depth sensor such as an RGBD camera or a laser range finder. Since a robot normally cannot perceive the whole environment

The authors are with the Robotics and Perception Group, University of Zurich, Switzerland – `http://rpg.ifi.uzh.ch` . This research was funded by the DARPA FLA Program, the National Center of Competence in Research (NCCR) Robotics through the Swiss National Science Foundation and the SNSF-ERC Starting Grant.

from one position, exploration reduces to repeatedly deciding where to move next at a given time when the area is already partially explored. This is essentially the same as the next-best-view (NBV) problem, which has been studied for reconstructing 3D objects with a sequence of depth scans for several decades [1], [2].

An adaptation of the NBV problem to a robotic context has been performed in [3]. It consists of adding two constraints: Firstly, a view may only be considered if a safely navigable path to it exists. Secondly, enough overlap between the current and next view must exist, such that the robot can register the two views. The proposed solution is to sample poses in the known free space, and to calculate for each sample a utility function that consists of the expected gain in terms of area that can be discovered and the cost to reach it. Then, from reachable samples that satisfy the overlap constraint, the one that maximizes the utility function is chosen. The cost is typically set to be proportional to the distance to the sample, but can also encode practical considerations such as minimizing behaviors that increase uncertainty [4]. In our approach, we assign a high cost to changes in velocity.

Frontier-based exploration is an alternative to NBV approaches that was proposed in [5]. The environment is discretized into a 2D or 3D grid, where each cell is labeled as occupied, free or unknown. Frontier cells are defined as free cells that are adjacent to unknown cells. Instead of sampling candidate views, frontier-based exploration assumes that simply navigating to a frontier will result in the exploration of new space. In [5], the robot navigates to the closest frontier found by depth-first search. Frontier-based exploration has become a popular exploration approach that has been extended in several publications, for example to support exploration using multiple robots [6], [7]. Extensions also exist for single robots systems, but it has been repeatedly shown that classic frontier-based exploration remains competitive [8] or even outperforms [9] these extensions. [8] also shows that the frontier-based approach outperforms the NBV approach presented in [3]. As a consequence, we will use as one baseline the nearest-frontier approach as implemented in [9]. For situations where uncertainty cannot be neglected, [9] shows that algorithms designed for that scenario, such as [10] or [11] result in a lower map error.

Exploration specifically using multi-rotors has been shown in [12], [13] and [14], among others. In [12], the authors propose a frontier-based exploration strategy for quadrotors using the 3D occupancy map OctoMap [15]. As in [8], this occupancy map is continuously updated, and thus more information is captured. Furthermore, before targeting a new frontier, flood-fill is used to reject unreachable frontiers, reducing time spent in trying to reach them. [13] proposes a novel way of determining frontier locations, which does not rely on explicitly representing the full space; only occupied space is represented. Rather than representing free and unknown space, particles are sampled within known free space. These particles are then used to simulate a gas that is contained by the known occupied space, and frontiers are identified as the places in which the gas expands. In terms of exploration speed, however, their experiments show that the frontier exploration approach by [7] performs better. [14] revisits NBV-based exploration. Instead of sampling poses in the free space, feasible trajectories are sampled using RRT* [16]. They do not execute a full trajectory, but rather the first part of the planning tree that is common to most trajectories with high utility. Then, the planning step is repeated in a receding horizon fashion. The authors compare their approach with [3] and show superior performance with faster calculation times and more explored space. We will use their publically available method as a second baseline for our comparisons.

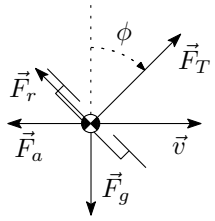

Fig. 2. Forces acting on the multi-rotor during horizontal flight.

III. Flight velocity for optimal energy use

An important motivation for our work is that the limited multi-rotor energy supply is not well used when flying near-hover. This is intuitive: when hovering, energy is used while no new ground is being covered. As we put more energy into motion, less is "wasted" on hovering. However, very large velocities are also inefficient due to aerodynamic drag. Thus, given a direction of motion, there must be an optimal velocity $v^\star$ at which most distance is traveled per energy use. Let us consider horizontal motion. The optimal velocity can be expressed as:

$$v^\star = \arg\max_v \frac{v}{P} \sim \arg\max_v \frac{v}{T} \tag{1}$$

where P is the power used. Since P is roughly proportional to the thrust T, we can P with T. To estimate a typical $v^\star$, we consider the multi-rotor model from [17]. We extend their translational dynamics model with an approximation of the aerodynamic drag of the quadrotor body, see Fig. 2:

$$\begin{aligned} m\vec{a} &= \vec{F}_T + \vec{F}_g + \vec{F}_r + \vec{F}_a \\ &= T\mathbf{R}\vec{e}_3 - mg\vec{e}_3 - Tk\mathbf{R}\vec{v}\tfrac{\vec{v}\cdot\mathbf{R}\vec{v}}{\vec{v}\cdot\vec{v}} - |\vec{v}|\vec{v}c_{\text{aero}}, \end{aligned} \tag{2}$$

with m the multi-rotor mass, $\vec{a}$ the acceleration, $\mathbf{R}$ the rotation matrix representing the multi-rotor attitude, $\vec{e}_3$ the unit vector in z-direction of the world frame, g the gravitational constant, k a first-order drag coefficient due to rotors, derived in [17] and c_{aero} the second-order drag coefficient due to the multi-rotor body. We make the assumption that c_{aero} is isotropic. In steady-state motion, $\vec{a} = 0$ and all forces act in a plane spanned by $\vec{v}$ and $\vec{e}_3$. Then, considering horizontal motion, (2) can be decomposed into a horizontal and vertical component:

$$\begin{aligned} c_{\text{aero}}v^2 + \cos^2\phi kTv &= \sin\phi T \\ \sin\phi\cos\phi kTv + \cos\phi T &= mg \end{aligned} \tag{3}$$

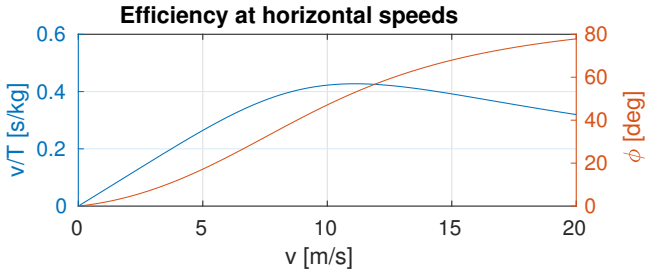

Fig. 3. Efficiency $\frac{v}{T}$ and corresponding pitch ϕ as a function of the horizontal velocity v calculated using the model (3). The parameters $m = 1.9\text{kg}$, $k = \frac{0.21}{gs}$ and $c_{\text{aero}} = 0.15$ are taken from the quadrotor specifications in [17]. While the values do not represent all multi-rotors, they give an idea of the magnitude of the optimal velocity.

where ϕ is the pitch angle. Using the multi-rotor specifications from [17], we solve this system of equations for different v and plot $\frac{v}{T}(v)$ in Fig. 3. As we can see, $v^\star$ is just above $10\frac{m}{s}$, with the pitch just above $50°$, which motivates flight at high speeds. In practice, there is an upper limit on the velocity v_{max} due to thrust limits, obstacle avoidance reaction time and speed limits of state estimation. We will henceforth assume $v^\star > v_{\text{max}}$ and consider v_{max} a parameter.

To enable effective obstacle avoidance, we align the depth sensor, for which we assume a limited field of view, with the flight direction $\vec{v}$ as much as is possible using only yaw.

IV. Methodology

We approximate the environment with a regular 3D voxel grid $V = \{\vec{x}\}$ with voxel dimensions s^3, where each voxel is represented by its centroid. Voxels are labeled free, occupied or unknown: $V = V_{\text{free}} \cup V_{\text{occ}} \cup V_{\text{unk}}$. The labeling is obtained from the robot's RGBD sensor measurements using OctoMap [15], assuming a perfect pose estimate but noisy depth measurements. We define the global set of frontiers

$$\mathcal{F}_g := \{\vec{x}_i \in V_{\text{free}} : \exists \vec{x}_j \in V_{\text{unk}} : |\vec{x}_i - \vec{x}_j| = s\}. \quad (4)$$

Previous frontier-based approaches typically decide where to move next by considering all $\mathcal{F}_g$. One approach is to run the Dijkstra algorithm [18], starting from the position of the robot and allowing transitions between adjacent $\vec{x} \in V_{\text{free}}$ until a frontier $\vec{x} \in \mathcal{F}_g$ is reached [9]. [14] samples possible trajectories using RRT* and moves towards the most promising direction. What can be observed with these approaches is that they require time to calculate, resulting in a stop-and-go behavior in which the robot moves for a while, stops to perform the calculation, then moves again. Furthermore, the goal frontier often lies behind the robot. A multi-rotor flying at a non-zero velocity would thus need to reverse its flight direction and re-visit ground that it has already covered. Evidently, all frontiers need to be visited sooner or later. But we find that choosing a direction and maintaining it, and only later returning, can result in faster coverage than going back and forth in a breadth-first-search manner, especially when flying at high speeds.

To avoid these problems, we propose to restrict the considered frontiers to those that are in the current field of view: $\mathcal{F}_v \subseteq \mathcal{F}_g$. This has three advantages: firstly, $\mathcal{F}_v$ can be calculated very efficiently from the operations that anyways need to be performed at every depth measurement. Secondly, the size of $\mathcal{F}_v$ is small enough that it allows a rapid decision and makes path planning inexpensive. Finally, any $\vec{x} \in \mathcal{F}_v$ will lie in flight direction of the multi-rotor, avoiding the effort to change the current velocity and avoiding flying back and forth, see Fig. 4. One could consider a different restriction, such as picking frontiers that lie in front of the robot, but that would require to perform a query in Octomap, whereas obtaining $\mathcal{F}_v$ can be easily combined with the most recent incorporation of depth measurements. Furthermore, $\mathcal{F}_v$ is guaranteed not to contain any frontiers that are currently occluded and thus not directly reachable. Note that $\mathcal{F}_v$ depends on the field of view of the robot - a broader field of view will result in a larger $\mathcal{F}_v$.

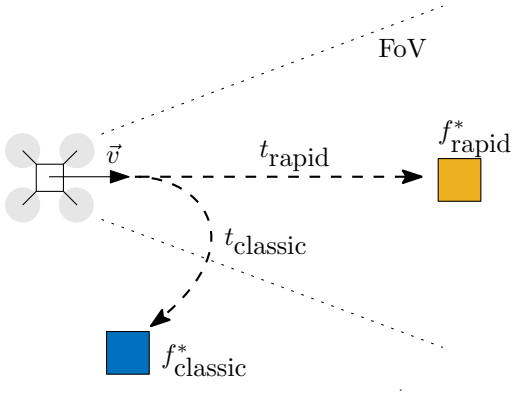

Fig. 4. The gist of our approach is that in frontier selection, we prefer frontiers which lie in flight direction. This allows continuous flight at high speeds and reduces the time spent flying through known areas.

From $\mathcal{F}_v$ we select the frontier $\vec{x}_i^\star$ with the lowest cost $c_i = |\vec{v}_i - \vec{v}_{\text{curr}}|$, the norm of the difference between the current velocity and the desired velocity at the frontier, $\vec{v}_i$. Recall from Section III that we on one hand aim to fly at v_{max} as much as possible. On the other hand, we want to have a velocity that allows us to react to yet unknown obstacles. Thus, we define the desired velocity for a frontier as

$$\vec{v}(\vec{x}_i) = (\vec{x}_i - \vec{x}_r) \cdot \frac{v_{\text{max}}}{r_{\text{sensor}}}, \quad (5)$$

$\vec{x}_r$ being the robot position. For frontiers that are at the detection range r_{sensor} of the depth sensor, the desired velocity will be v_{max} and pointing towards the unknown volume, while for frontiers closer to the robot the desired velocity will be lower. This imposes a slower approach of frontiers behind which there might be nearby obstacles.

Once we have determined $\vec{x}_i^\star$, we set the robot velocity to $\vec{v}_i$. Lest the robot tries to reach a physically unreachable frontier, we further restrict the choice of frontiers to the ones that are accessible from the current position:

$$\mathcal{F}_a = \{\vec{x} \in \mathcal{F}_v : \mathcal{A}(\vec{x})\} \quad (6)$$

$$\mathcal{A}(\vec{x}) : \nexists \vec{x}_{occ} \in V_{\text{occ}}, \vec{x}_s \in S_r(\vec{x}) : |\vec{x}_{occ} - \vec{x}_s| < d_{\text{safe}} \quad (7)$$

with $S_r(\vec{x})$ the set of points on the segment between the current position and the frontier $\vec{x}$ and d_{safe} a minimum safety radius (see Fig. 5). The control loop runs every $20ms$.

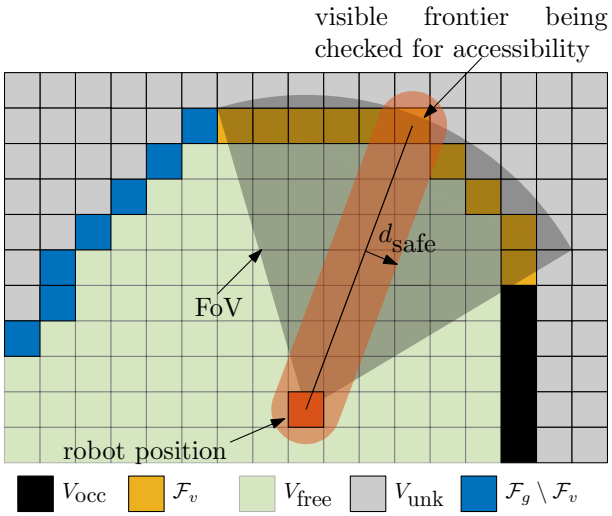

Fig. 5. Two dimensional illustration of determining the accessibility of members of the visible frontier set.

As soon as $\mathcal{F}_a = \emptyset$, the algorithm switches to a classical frontier selection method, which is implemented as discussed in [9]: a shortest path with waypoints $W(\vec{x}_i) = \{\vec{w}_1, \vec{w}_2, ..., \vec{w}_m\}$ to any frontier $\vec{x}_i \in \mathcal{F}_g$ is sought using the Dijkstra algorithm. If no such path is found, exploration is considered complete. Otherwise, the robot will track W. This path tracking is aborted if one of two conditions occurs: if $\vec{w}_m$ has been reached or is unreachable, the frontier $\vec{x}_i$ that had generated the path is removed from $\mathcal{F}_g$ and a new path W is calculated. Alternatively, if at any point of the waypoint tracking $\mathcal{F}_a \neq \emptyset$, the control switches back to the rapid frontier selection method.

In order to fly close to $v_{\max}$ during W tracking, the multi-rotor velocity is set such that it flies at the fastest safe speed towards the furthest accessible waypoint:

$$\vec{v}_r \leftarrow \vec{v}(\arg\max_{\vec{w}_j} j : \mathcal{A}(\vec{w}_j)) \tag{8}$$

where $\vec{v}(\vec{x})$ is the same as in (5), with the norm truncated to $v_{\max}$.

V. Experiments

In order to evaluate and compare the performance of the proposed approach, simulation studies have been performed using RotorS [19] and Gazebo. Furthermore, a real world experiment has been conducted to demonstrate the rapid exploration algorithm on a real quadrotor. In all experiments we use the control architecture from [20].

A. Simulation

The simulated multi-rotor is equipped with a depth camera mounted in a forward-looking configuration. The stereo camera has a field of view (FoV) of [60, 115]° in vertical and horizontal direction. The proposed algorithm (*"Rapid"*) is compared with the classic frontier-based exploration as implemented in [9] (*"Classic"*) as well as with the next-best-view approach presented in [14] (*"NBV"*). To make a fair comparison, we have tried to make the quadrotor fly at the same maximum speed $v_{\max}$ for all approaches, as far as this is possible with the selected trajectories. To that end, we employ the same waypoint-based navigation for *Classic* as discussed in Section IV. *NBV* is limited to low speeds and we were not able to deploy it at the full range of velocities that we evaluate. The reason for this speed limitation is the way the quadrotor navigates to its new goal. Instead of generating a smooth trajectory, the quadrotor attempts to navigate to the next node of the generated random tree in a straight line. Since this results in instantaneous changes of direction at every node of the tree, for velocities above 0.7 meters per second the quadrotor fails to follow the trajectory. This often results in a crash into nearby obstacles. Furthermore, the maximum yaw rate was limited to 0.75 rad/s for *NBV* for the same reasons.

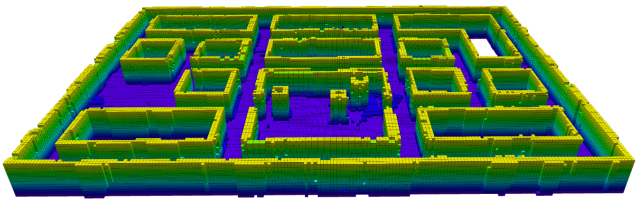

Fig. 6. Fully explored Juliá scenario, with dimensions 38x26x3 m. For visualization reasons, the map is truncated at a height of 2.5 meters.

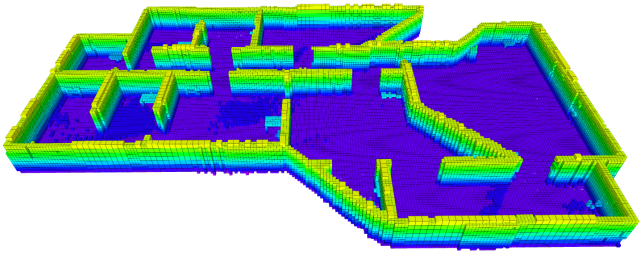

Fig. 7. Fully explored Office scenario, with dimensions 38x23x3 m. For visualization reasons, the map is truncated at a height of 2.5 meters.

We perform simulations on three different scenarios: **"Juliá"**, an environment mimicking *Scenario 2* from [9], see Fig. 6. This 2D scenario is dominated by corridors and contains two open spaces. **"Office"**, another 2D scenario with a more open, office-like layout as depicted in Fig. 7. And finally **"Powerplant"**, a 3D scenario of a powerplant obtained from the Gazebo model library[1]. The original powerplant model is cropped to smaller dimensions as depicted in Fig. 8. Specific parameters for the scenarios are listed in tables I and II. Parameters $d_{\max}^{\text{planner}}$, λ, $N_{\max}$ and the maximum edge length of the RRT tree refer to the setup of *NBV* and are explained in [14]. Since exploration performance can

[1] https://bitbucket.org/osrf/gazebo_models/src

Parameter	Value	Parameter	Value
$v_{\max}$	$\{0.3, 0.7, 1.5, 2.5\}$m/s	FoV	[60x115]°
Resolution	0.2m	$\dot{\varphi}_{\max}$	1.5rad/s
d_{safe}	0.6m	$\dot{\varphi}_{\max}^{\text{NBV}}$	0.75rad/s

TABLE I

Parameters common in the different simulation scenarios.

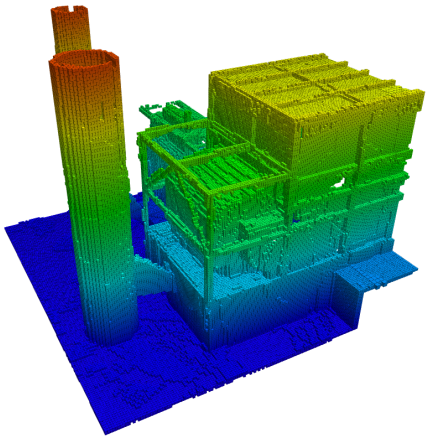

Fig. 8. Fully explored Powerplant scenario, with dimensions 33x31x26 m.

	Juliá	**Office**	**Powerplant**
Dimensions [m]	38x26x3	38x23x3	33x31x26
$\mathbf{d}_{\text{max}}^{\text{sensor}}$ [m]	5.0	5.0	7.0
$\mathbf{d}_{\text{max}}^{\text{planner}}$ [m]	1.5	1.5	2.0
λ	0.3	0.2	0.2
$\mathbf{N}_{\text{max}}$	20	20	30
RRT max edge length [m]	1.0	1.0	3.0

TABLE II

PARAMETERS CHANGING FOR THE DIFFERENT SCENARIOS.

be dependent on the initial position of the robot, simulations were performed for multiple initial positions. Furthermore, the maximum velocity of the quadrotor was varied and a complete set of six simulations was performed for each velocity. Note that our method is defined in 3D. Thus, the 2D scenarios are extruded to a height of $2.5m$ and closed with floor and ceiling. The height is low enough so that the area can be covered while flying at a single altitude.

B. Real World Experiments

To verify that the speeds achieved in simulation can also be reached in the real world, we implemented our exploration algorithm on a real quadrotor, and made it explore both an indoor and an outdoor scenario: The indoor scenario is a room with dimensions 6.5x6.8x2.6 meters, see Fig. 9, and the outdoor scenario is a forest, see Fig. 10. Since there were no natural bounds in the outdoor scenario, we artificially restricted the motion of the quadrotor to a bounding box expressed relative to its starting point. This results in an inconsistent amount of free space between different runs, and we had to change the experiment location frequently due to lighting conditions, so we only present quantitative results for the indoor scenario. The depth sensor used for both real world scenarios was the Intel RealSense R200 with a FoV of [56,43] degrees in horizontal and vertical direction, respectively, and was mounted in a forward-looking configuration at a pitch angle of 0 degrees. A summary of the parameters applied in the indoor scenario is given in Table III. Note that the sensor range is set at different values that are all below the actual range of the sensor. We have done this to require the

Fig. 9. Quadrotor flying in the indoor real world scenario.

Fig. 10. Quadrotor flying in the outdoor real world scenario.

quadrotor to move – had we used the full sensor range, the quadrotor would not have needed to move very far to explore the room at hand. At higher velocities, however, we need to increase the sensor range such that the quadrotor is able to react to obstacles. The outdoor experiments were performed at $\{1, 1.5, 2\}\frac{m}{s}$ and with $r_{sensor} = 5m$. For state estimation, we use the pipeline described in [20], which relies on visual odometry by SVO [21] that is fused with an IMU estimate using MSF [22].

C. Measurements

In each experiment, the count of cells currently estimated free $|V_{\text{free}}|(t)$, the state at which the robot is in (rapid exploration, calculating W or tracking W) as well as the current position and velocity of the quadrotor are sampled at $5Hz$. From this, the coverage ratio $\frac{|V_{\text{free}}|}{|V_{\text{free}}^{\star}|}(t)$, where $V_{\text{free}}^{\star}$ is the actual free space, and the distance traveled $d_{\text{tot}}(t)$ are measured for all samples at times t. To compare the approaches, we report as main performance metric the time at which the scenario is fully explored t_{max}, the total distance

Parameter	**Value**	**Parameter**	**Value**
v_{max}	$\{0.3, 0.7, 1, 1.5, 2\}$m/s	FoV	[43x56]°
Resolution	0.2m	$\dot{\varphi}_{\text{max}}$	1.5rad/s
r_{sensor}	$\{3, 3.5, 4\}$ m	d_{safe}	0.7m

TABLE III

PARAMETERS USED FOR THE INDOOR EXPERIMENT.

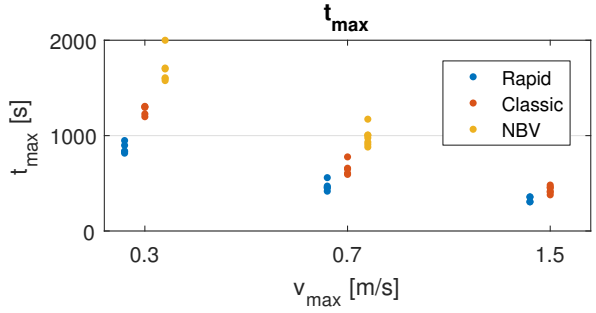

Fig. 11. Exploration times in office scenario. Each dot represents one of six runs.

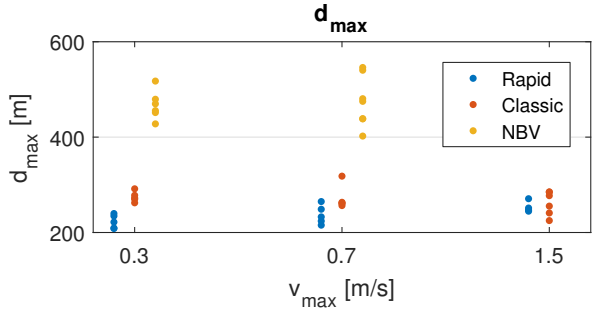

Fig. 12. Traveled path in office scenario. Each dot represents one of six runs.

traveled $d_{\max}$ and the expected voxel discovery time

$$t_{\text{exp}} = \frac{1}{|V^{\star}_{\text{free}}|} \sum_{\vec{x} \in V^{\star}_{\text{free}}} \min\{t : \vec{x} \in V_{\text{free}}(t)\}. \tag{9}$$

The last metric is more meaningful than $t_{\max}$ for scenarios where the goal is to find several objects in an unknown environment as quickly as possible (see related discussion in [9]).

We have found that with our implementation of Dijkstra, the robot often spends a significant amount of time calculating the path W. Especially in later stages of the exploration process, where the path to the next frontier tends to become longer, the path calculation can take up to 10 seconds. In order to estimate only the quality of the resulting flight behavior, we adjust for this calculation time by not counting time spent on it when calculating the above performance metrics. This mainly benefits $t_{\max}$ and t_{mean} of the classic frontier approach. The calculation time of both the reactive mode and of the NBV method are negligible, and so we do not need to adjust for them.

VI. Results

A. Simulation

The main results are reported in Table IV. For the office scenario, the comparison of the main performance metrics for different values of $v_{\max}$ are visualized in Figs. 11, 12 and 13. As can be seen, our approach consistently outperforms classical Frontier-based exploration and next-best-view exploration across all scenarios with respect to $t_{\max}$. The improvement is even more significant for t_{exp}, as in our approach there is often a significant amount of

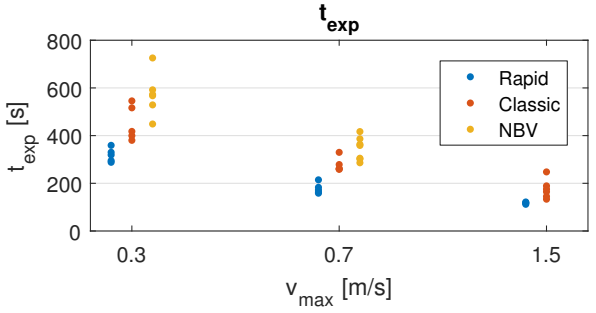

Fig. 13. Expected voxel exploration time for office scenario. Each dot represents one of six runs.

time at the end used only for re-visiting frontiers that have been previously skipped. This behavior is well illustrated in Fig. 1, which shows the evolution of $\frac{|V_{\text{free}}|}{|V^{\star}_{\text{free}}|}(t)$ and $d_{\text{tot}}(t)$ for one instance of *Rapid* and *Classic* each in scenario Juliá, $v_{\max} = 2.5$. Within the approaches, $t_{\max}$ and t_{exp} decrease as $v_{\max}$ increases, as can be expected. However, as our approach is designed for high speeds, its decrease in $t_{\max}$ and t_{exp} is most significant, in particular in the Juliá and Powerplant scenarios.

We are surprised by the poor results that we obtained from NBV, in spite of parameter tuning. In particular, in Juliá it consistently gets stuck in one part of the map and does not move towards the unexplored regions. Thus, we cannot report any results for that scenario. For the scenarios in which it did succeed, its performance was mostly inferior to the other two approaches. This is consistent with the results in [8], which shows longer exploration times for a previous NBV-based approach [3], compared to a classic frontier-based approach. In Fig. 14 it can be seen that the NBV trajectory is less smooth than the trajectories performed by the other two approaches.

Of particular interest is the behavior of $d_{\max}$. Within the runs of both the NBV planner and classic frontier-based exploration, $d_{\max}$ is not significantly affected by $v_{\max}$. For the proposed exploration algorithm, however, it increases with $v_{\max}$. We assume that this happens because a higher $v_{\max}$ will cause the multi-rotor both to overshoot in dead ends and to make wider turns. The comparison of $d_{\max}$ between Rapid and Classic Frontier exploration is not consistent. For Juliá and Powerplant, $d_{\max}$ is generally higher for our approach than for Classic Frontier exploration. However, $d_{\max}$ is lower for our approach in the Office scenario. Here, we discern two effects: on one hand, our approach does not minimize distance traveled as does the Classic Frontier-based approach. On the other hand, our approach, which is more reactive, results in smoother trajectories. As can be seen in Fig. 14, this is beneficial in the Office scenario. In a scenario which is more cluttered and which imposes more narrow turns, we would expect a smaller difference in overall performance between our approach and classic frontier based exploration.

Another observation that can be made is that classical frontier based exploration outperforms NBV exploration in

		Rapid			NBV Planner			Classic Frontier		
Scenario	$\mathbf{v}_{\max}[\frac{m}{s}]$	$\mathbf{d}_{\max}[m]$	$\mathbf{t}_{\max}[s]$	$\mathbf{t}_{\exp}[s]$	$\mathbf{d}_{\max}[m]$	$\mathbf{t}_{\max}[s]$	$\mathbf{t}_{\exp}[s]$	$\mathbf{d}_{\max}[m]$	$\mathbf{t}_{\max}[s]$	$\mathbf{t}_{\exp}[s]$
Julia	**0.3**	262 ± 8.9	1074 ± 55	379 ± 31				268 ± 24	1261 ± 124	490 ± 34
	0.7	273 ± 15	553 ± 59	202 ± 13				258 ± 14	636 ± 28	264 ± 15
	1.5	310 ± 28	408 ± 49	121 ± 15				263 ± 19	531 ± 78	221 ± 35
	2.5	315 ± 30	360 ± 39	105 ± 8.7				260 ± 26	505 ± 58	211 ± 14
Office	**0.3**	223 ± 14	866 ± 55	318 ± 29	466 ± 30	1698 ± 158	573 ± 91	275 ± 11	1266 ± 50	452 ± 74
	0.7	237 ± 20	471 ± 53	178 ± 22	474 ± 54	983 ± 96	346 ± 49	272 ± 26	657 ± 73	278 ± 30
	1.5	253 ± 12	332 ± 29	117 ± 4.1				261 ± 25	433 ± 38	176 ± 41
Powerplant	**0.7**	692 ± 53	1245 ± 151	364 ± 31	1363 ± 290	2104 ± 406	613 ± 131	692 ± 57	2397 ± 170	852 ± 86
	1.5	710 ± 65	717 ± 94	198 ± 2.1				692 ± 32	1519 ± 88	567 ± 31
	2.5	728 ± 49	582 ± 26	150 ± 3.9				684 ± 56	1437 ± 123	565 ± 80

TABLE IV

MEAN AND STANDARD DEVIATION FOR THE TOTAL DISTANCE TRAVELED s_{MAX}, THE TOTAL TIME SPENT EXPLORING t_{MAX} AND THE EXPECTED CELL DISCOVERY TIME t_{EXP} ACCROSS ALL EXPERIMENTS.

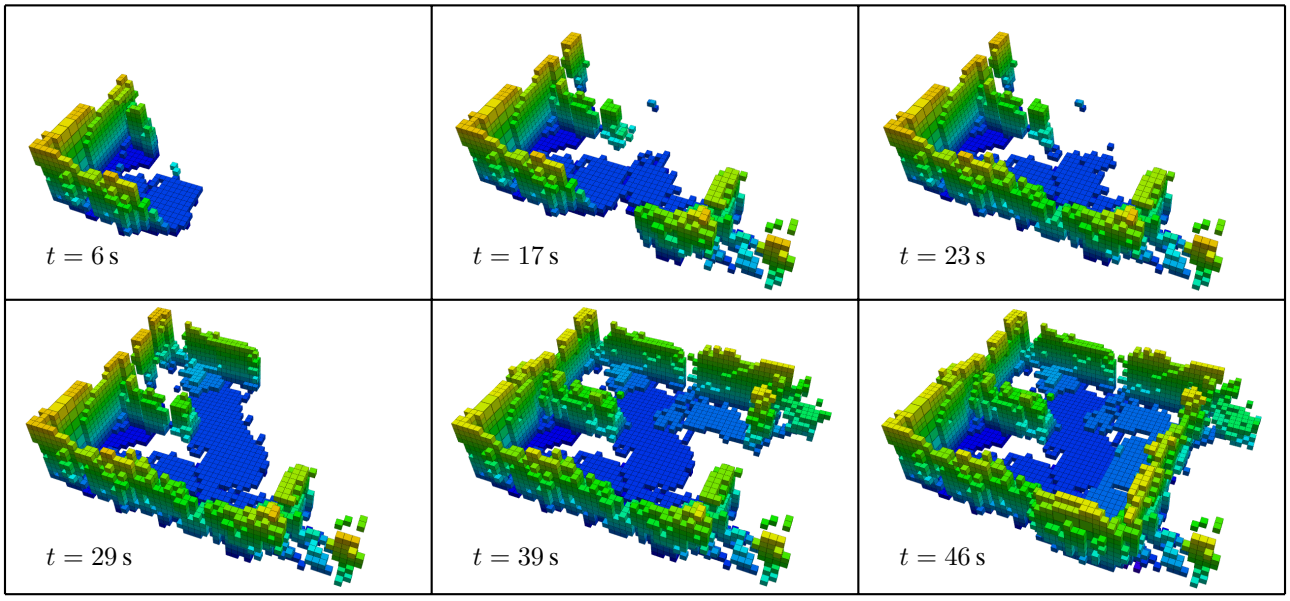

Fig. 15. Real world exploration of an empty room that is halfway separated by a wall. The top speed of the quadrotor is $2\frac{m}{s}$.

the two dimensional case of office exploration. In the 3D case, however, classic frontier based exploration shows poor performance. A reason for this drop in performance is the limited field of view of the sensor. While in the 2D case, the sensor usually manages to simultaneously detect floor and ceiling at the same time, in the large 3D environment this is no longer the case. As a result, the quadrotor selects frontiers on the border of the view frustum of the camera and moves step-wise up or down, without large movements in the xy-plane. The NBV controller is better able to handle this and outperforms the classical frontier-based method. Rapid exploration, however, again shows the best performance in the 3D case, since the quadrotor will fly with the maximum velocity for long distances.

B. Real World Experiments

The quadrotor successfully managed to map the real world scenarios at the tested speeds. Table V shows the performance of the proposed approach on the indoor scenario

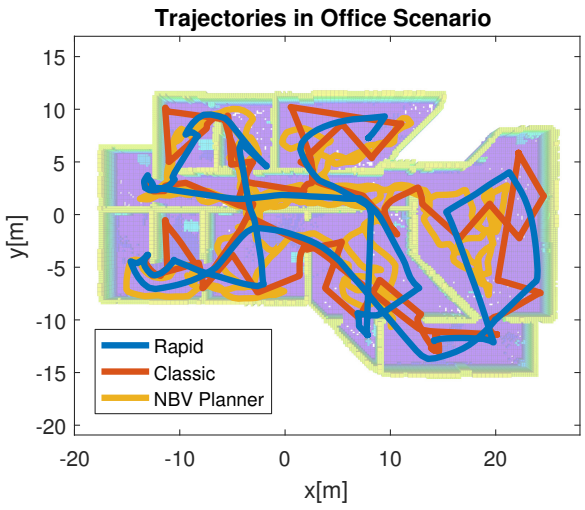

Fig. 14. Comparison of trajectories in Office scenario.

	Rapid		
$\mathbf{v}_{\max}[\frac{\mathbf{m}}{\mathbf{s}}]$	$\mathbf{d}_{\max}[m]$	$\mathbf{t}_{\max}[s]$	$\mathbf{t}_{\exp}[s]$
0.3	29.4	125.7	42.6
0.5	21.3	66.9	26.2
0.7	27.7	77.6	27.8
1.0	24.1	63.3	23.5
1.5	26.4	54.4	19.9
2.0	28.7	52.5	23.6

TABLE V

TOTAL DISTANCE TRAVELED d_{MAX}, THE TOTAL TIME SPENT EXPLORING t_{MAX} AND THE EXPECTED CELL DISCOVERY TIME t_{EXP} OBTAINED FOR THE INDOOR REAL WORLD EXPERIMENT.

with respect to the three metrics for various $v_{\max}$. Apart from $v_{\max} = \{0.3, 0.7\}\frac{m}{s}$, the total exploration time $t_{\max}$, as well as the expected exploration time $t_{\exp}$ decrease with larger velocities. Since only one run was performed for each velocity, we consider $v_{\max} = \{0.3, 0.7\}\frac{m}{s}$ to be outliers.

Most of the failures that were experienced in the real world can be attributed to one of three causes: firstly, pose estimation failures, particularly with rapid changes in attitude above terrain. This is due to the visual odometry being based on the image of a down-looking camera, whose image would rapidly change. To prevent such failures indoors, we set the room up such that attitude above terrain changes would be minimal. Outdoors, we chose locations dominated by tree trunks, with little vegetation on the ground. Secondly, the RealSense sensor would sometimes fail to see objects, in particular the curtain that we had on one side of the room, which would rapidly flap as the quadrotor would approach it. Clamping the bottom of the curtain significantly decreased the frequency at which this problem occurred. Thirdly, the geometry of the sensor placement and field of view resulted in a blind spot for obstacles at certain low-radius turns executed in the reactive mode. This problem was solved by artificially restricting the field of view when considering the set of visible frontiers $\mathcal{F}_v$.

Fig. 15 depicts the indoor exploration process at six distinct time instances and illustrates the growth of the map representing the environment at $v_{\max} = 2\frac{m}{s}$. Note that after starting the exploration task, the quadrotor waits for 3 seconds to assure a sufficient OctoMap representation of the environment before it starts to navigate to frontiers. For more insights, we invite the reader to take a look at the multimedia material at `https://youtu.be/54s6gGZLpJo`.

VII. CONCLUSION

Within this work, an exploration algorithm was proposed that is designed specifically for multi-rotor exploration at high speeds. The reactive behavior of the algorithm allows for fast incorporation of new information and results in efficient trajectories. Compared to classic frontier-based exploration, the approach can occasionally exhibit a small increase in the total path traveled to explore an area, but at the same time achieves smaller exploration times for the same maximum velocity constraint. These properties are demonstrated in multiple simulations for different exploration scenarios and validated with real world experiments.

REFERENCES

[1] C. Connolly, "The determination of next best views," in *Robotics and Automation. Proceedings. 1985 IEEE International Conference on*, vol. 2. IEEE, 1985, pp. 432–435.

[2] J. Maver and R. Bajcsy, "Occlusions as a guide for planning the next view," *IEEE transactions on pattern analysis and machine intelligence*, vol. 15, no. 5, pp. 417–433, 1993.

[3] H. H. Gonzalez-Banos and J.-C. Latombe, "Navigation strategies for exploring indoor environments," *The International Journal of Robotics Research*, vol. 21, no. 10-11, pp. 829–848, 2002.

[4] B. Tovar, L. Muñoz-Gómez, R. Murrieta-Cid, M. Alencastre-Miranda, R. Monroy, and S. Hutchinson, "Planning exploration strategies for simultaneous localization and mapping," *Robotics and Autonomous Systems*, 2006.

[5] B. Yamauchi, "A frontier-based approach for autonomous exploration," in *Computational Intelligence in Robotics and Automation, 1997. CIRA'97., Proceedings., 1997 IEEE International Symposium on*. IEEE, 1997, pp. 146–151.

[6] ——, "Frontier-based exploration using multiple robots," in *Proceedings of the second international conference on Autonomous agents*. ACM, 1998, pp. 47–53.

[7] W. Burgard, M. Moors, C. Stachniss, and F. E. Schneider, "Coordinated multi-robot exploration," *IEEE Transactions on robotics*, vol. 21, no. 3, pp. 376–386, 2005.

[8] D. Holz, N. Basilico, F. Amigoni, and S. Behnke, "Evaluating the efficiency of frontier-based exploration strategies," *Isr/Robotik 2010*.

[9] M. Juliá, A. Gil, and O. Reinoso, "A comparison of path planning strategies for autonomous exploration and mapping of unknown environments," *Autonomous Robots*, vol. 33, no. 4, pp. 427–444, 2012.

[10] A. A. Makarenko, S. B. Williams, F. Bourgault, and H. F. Durrant-Whyte, "An experiment in integrated exploration," in *Intelligent Robots and Systems, IEEE/RSJ International Conference on*, 2002.

[11] M. Juliá, O. Reinoso, A. Gil, M. Ballesta, and L. Payá, "A hybrid solution to the multi-robot integrated exploration problem," *Engineering Applications of Artificial Intelligence*, 2010.

[12] 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 *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, 2012.

[13] S. Shen, N. Michael, and V. Kumar, "Stochastic differential equation-based exploration algorithm for autonomous indoor 3d exploration with a micro-aerial vehicle," *The International Journal of Robotics Research*, vol. 31, no. 12, pp. 1431–1444, 2012.

[14] A. Bircher, M. Kamel, K. Alexis, H. Oleynikova, and R. Siegwart, "Receding horizon "next-best-view" planner for 3d exploration," in *2016 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2016, pp. 1462–1468.

[15] A. Hornung, K. M. Wurm, M. Bennewitz, C. Stachniss, and W. Burgard, "Octomap: An efficient probabilistic 3d mapping framework based on octrees," *Autonomous Robots*, 2013.

[16] S. Karaman and E. Frazzoli, "Sampling-based algorithms for optimal motion planning," *The International Journal of Robotics Research*, vol. 30, no. 7, pp. 846–894, 2011.

[17] S. Omari, M.-D. Hua, G. Ducard, and T. Hamel, "Nonlinear control of vtol uavs incorporating flapping dynamics," in *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, 2013.

[18] E. W. Dijkstra, "A note on two problems in connexion with graphs," *Numerische mathematik*, vol. 1, no. 1, pp. 269–271, 1959.

[19] F. Furrer, M. Burri, M. Achtelik, and R. Siegwart, *RotorS—A Modular Gazebo MAV Simulator Framework*. Springer International Publishing, 2016, pp. 595–625.

[20] M. Faessler, F. Fontana, C. Forster, E. Mueggler, M. Pizzoli, and D. Scaramuzza, "Autonomous, vision-based flight and live dense 3d mapping with a quadrotor micro aerial vehicle," *Journal of Field Robotics*, vol. 4, no. 33, pp. 431–450, 2016.

[21] C. Forster, M. Pizzoli, and D. Scaramuzza, "Svo: Fast semi-direct monocular visual odometry," in *2014 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2014, pp. 15–22.

[22] S. Lynen, M. W. Achtelik, S. Weiss, M. Chli, and R. Siegwart, "A robust and modular multi-sensor fusion approach applied to mav navigation," in *Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on*. IEEE, 2013, pp. 3923–3929.