2016 IEEE International Conference on Robotics and Automation (ICRA)
Stockholm, Sweden, May 16-21, 2016

Receding Horizon "Next–Best–View" Planner for 3D Exploration

Andreas Bircher[1], Mina Kamel[1], Kostas Alexis[2], Helen Oleynikova[1] and Roland Siegwart[1]

***Abstract*— This paper presents a novel path planning algorithm for the autonomous exploration of unknown space using aerial robotic platforms. The proposed planner employs a receding horizon "next–best–view" scheme: In an online computed random tree it finds the best branch, the quality of which is determined by the amount of unmapped space that can be explored. Only the first edge of this branch is executed at every planning step, while repetition of this procedure leads to complete exploration results. The proposed planner is capable of running online, onboard a robot with limited resources. Its high performance is evaluated in detailed simulation studies as well as in a challenging real world experiment using a rotorcraft micro aerial vehicle. Analysis on the computational complexity of the algorithm is provided and its good scaling properties enable the handling of large scale and complex problem setups.**

I. Introduction

As mobile robots demonstrate higher levels of autonomy and can be used as a tool even by people without special training, the number of use cases increases rapidly. In applications like industrial inspection [1], [2], crop monitoring [3], search and rescue [4] or reef mapping [5], [6], mobile robots are designed to gather information autonomously, quickly and with high precision. At the same time their low cost allows the scheduling of frequent inspection or surveillance, while the use of robots in dangerous environments can reduce the risks humans are exposed to.

In the literature algorithms have been proposed, that can make plans for such missions [7], [8] assuming full prior knowledge about the environment and precompute good paths satisfying the mission requirements, while others do not assume any a priori information [9], [10] and compute paths online in reaction to the perceived environment. The latter are especially useful in changing or unknown environments, where paths cannot reliably be precomputed. Obstacles can obstruct the path and targets may have been moved to different locations, requiring such algorithms to replan during the mission, relying on local path planning schemes.

The approach described within this work aims to solve the problem of exploring an unknown 3D space. It proposes the use of a receding horizon planning strategy, within which an optimized exploration path (a sequence of steps and viewpoint configurations) is computed, but each time only the first step is executed, while the whole process is then repeated. The environment is mapped in an occupancy map [11] constructed from depth images input. In the resulting known free space, a finite iteration random tree is grown using a suitable tree construction algorithm (e.g. RRT [12], RRT* [13]) and each branch is evaluated for the amount of unmapped space that can be mapped. The *first edge* of the *best branch* is then executed. Thorough evaluation of the resulting performance using a simulated hexacopter Micro Aerial Vehicle (MAV) is provided, followed by a validation in an experiment using a real hexacopter MAV with a stereo camera module mounted. These results, complemented by a computational complexity analysis, reveal the high performance of the planner, exploring and navigating through complex 3D environments, while being able to run onboard the aerial robot. The used implementation is released as an open source code package [14].

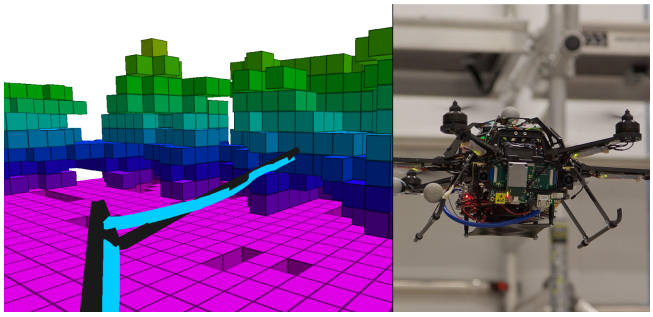

Fig. 1: An instant of the exploration of a room containing a scaffold. The right side depicts the employed hexacopter micro aerial vehicle, while the online dense reconstruction acquired during the exploration mission is shown on the left side. The computed path is visualized in black, closely followed by the MAV track in light blue.

The remainder of this paper is organized as follows: Section II overviews the related work and how our approach contributes further, while the considered problem is defined in Section III. The proposed solution is described in detail in Section IV and evaluated in simulation and real–world experiments in Sections V and VI. Finally, Section VII summarizes and concludes the presented work.

II. Related Work

Within the literature, various contributions have been made towards autonomous exploration. Early work includes [15], where good "next–best–views" are determined in order to cover a given structure. An advanced version of this iterative approach was presented in [9]. A previously unknown 2D area is explored in [16], considering a robotic system with

This work has received funding from the European Union's Horizon 2020 Research and Innovation Programme under the Grant Agreement No.644128, AEROWORKS.

[1]Andreas Bircher, Mina Kamel, Helen Oleynikova and Roland Siegwart are with the Autonomous Systems Lab, ETH Zurich, Leonhardstrasse 21, 8092 Zurich, Switzerland andreas.bircher@mavt.ethz.ch

[2]Kostas Alexis is with the University of Nevada, Reno, 1664 N. Virginia Street, Reno, NV 89557, USA

measurement and navigation errors. It employs hierarchical maps to describe the topological objects encountered and derives a distinctiveness map in which the controller steers "up–hill". In [17] the notion of frontier regions was introduced for the first time. They consist in regions in the exploration space, where known free space neighbours unmapped space. The mentioned paper proposes an algorithm that finds paths to poses at these frontiers that maximize the extension of the horizon of known space considering an onboard sensor. Repetition of this process leads to exploration of the whole space. Advanced variants of this algorithm were presented in [18], [19] and in [20], also improving the coverage of unknown space along the path to the frontier. The concept has been extended to multi–agent exploration e.g. in [21], [22]. Using a geometric representation of the detected obstacles, [10] samples possible viewpoints in the whole known free space instead of just at the frontiers. Finally, the work in [6] uses fuzzy logic and utility fusion concepts to autonomously explore coral reefs with underwater vehicles.

The approach proposed within this work also samples "next–best–views" in the known free space. In contrast to previous contributions, the views are sampled as nodes in a random tree, the edges of which directly give a path to follow in order to reach the viewpoint. In most iterations very few sampled viewpoints suffice to determine a reasonably good next step. The robot then only executes the first edge of the tree towards the best viewpoint, after which the process is repeated in a receding horizon fashion, which further distinguishes the presented approach from previously known exploration planning algorithms.

III. Problem Description

The problem considered within this work consists in exploring a bounded 3D space $V \subset \mathbb{R}^3$. This is, to determine which parts of the initially unmapped space $V_{unm} \overset{init.}{=} V$ are free $V_{free} \subset V$ or occupied $V_{occ} \subset V$. The operation is subject to vehicle dynamic constraints, localization uncertainty and limitations of the employed sensor system with which the space is explored. As for most sensors the perception stops at surfaces, hollow spaces or narrow pockets can sometimes not be explored with a given setup. This residual space is denoted by V_{res}. The problem is considered fully solved when $V_{free} \cup V_{occ} = V \setminus V_{res}$. Due to the nature of the problem, a suitable path has to be computed online and in real time, as free space to navigate in is not known prior to its exploration. To increase the autonomy of the vehicle, the planner should run onboard with the limited resources available.

IV. Proposed Approach

Considering the exploration of unknown space, the proposed approach employs a sampling–based receding horizon path planning paradigm, to generate paths that cover the whole space V. A sensing system that perceives an area of the environment, e.g. depth camera or a laser scanner is employed to provide feedback on the explored space. All acquired information is combined in a map representing the world. This map is used for both, collision free navigation and determination of the exploration progress. The structure of this closed loop setup, depicted in Figure 2, resembles receding horizon controllers like [23], [24]. In contrast to the mentioned controllers, the objective of the presented planner focuses on exploration, while penalizing distance. The continuous feedback by mapping the environment helps attenuating errors in tracking and perception, similar to control theory, where receding horizon concepts also improve robustness to such errors and uncertainties.

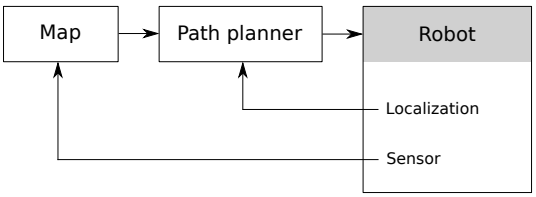

Fig. 2: Diagram of the system setup. A map of the environment is built using the input of onboard sensors of the robot. The mapped knowledge of the environment enables the planner to find good exploration paths starting at the current location that are then executed by the robot.

The employed representation of the environment is an occupancy map [11] dividing space V in cubical volumes $m \in \mathcal{M}$, that can either be marked as free, occupied or unmapped. The resulting array of voxels is saved in an octree, enabling computationally efficient access for operations like integrating new sensor data or checking for occupancy. Paths are only planned through known free space V_{free}, thus providing collision–free navigation. While a collision free vehicle configuration is denoted by $\xi \in \Xi$, a path is given by $\sigma : \mathbb{R} \to \xi$, specifically from ξ_{k-1} to ξ_k by $\sigma_{k-1}^k(s)$, $s \in [0,1]$ with $\sigma_{k-1}^k(0) = \xi_{k-1}$ and $\sigma_{k-1}^k(1) = \xi_k$. The path has to be such that it is collision free and can be tracked by the vehicle, considering its dynamic and kinematic constraints. The corresponding path cost is $c(\sigma_{k-1}^k)$. For a given occupancy map representing the world $\mathcal{M}$, the set of visible and unmapped voxels from configuration ξ is denoted as $\mathbf{Visible}(\mathcal{M}, \xi)$. Every voxel m in this set lies in the unmapped exploration area V_{unm}, the direct line of sight does not cross occupied voxels and in addition it complies with the sensor model (e.g. inside Field of View (FoV), maximum range). Starting from the current configuration of the vehicle ξ_0, a geometric tree $\mathbb{T}$ is incrementally built in the configuration space, within the scope of this work using the RRT algorithm [12]. The resulting tree contains $N_{\mathbb{T}}$ nodes n and its edges are given by collision–free paths σ, connecting the configurations of the two nodes. The quality - or collected information gain - of a node $\mathbf{Gain}(n)$ is the summation of the unmapped volume that can be explored at the nodes along the branch, e.g. for node k, according to

$$\mathbf{Gain}(n_k) = \mathbf{Gain}(n_{k-1}) + \mathbf{Visible}(\mathcal{M}, \xi_k) e^{-\lambda c(\sigma_{k-1}^k)}, \quad (1)$$

with the tuning factor λ penalizing high path costs [18].

After every replanning, the first segment of the branch to the best node $\mathbf{ExtractBestPathSegment}(n_{best})$ is

executed by the robot, where n_{best} is the node with the highest gain. To prevent discarding of already found high quality paths, the remainder of the best branch is used to reinitialize the tree in the next planning iteration, re–evaluating its gain using the updated map $\mathcal{M}$. Tree creation is in general stopped at $N_{\mathbb{T}} = N_{\max}$, but if the best gain g_{best} of n_{best} remains zero, the tree construction is continued, until $g_{best} > 0$. Maintaining the best branch for the next planning steps prevents the loss of these computationally expensive paths. For practical purposes, a tolerance value N_{TOL} is chosen significantly higher than $N_{\max}$ and the exploration is considered solved when $N_{\mathbb{T}} = N_{TOL}$ is reached, while g_{best} remained zero. Algorithm 1 summarizes the planning procedure.

A. Implementation Details

The described approach was designed to equip an aerial robot with exploration autonomy but is also applicable to a wide range of robot configurations including Unmanned Ground Vehicles (UGV) and Autonomous Underwater Vehicles (AUV). Path generation and tree construction algorithms are formulated in a general manner, which allows the use of any suitable implementation. In particular, any nonholonomic constraints can conveniently be implemented when growing the tree (e.g. by sampling inputs and subsequent forward integration of the vehicle model). The actual implementation used for the experiments presented in this work is designed to plan for a rotorcraft MAV.

The considered vehicle configuration is the flat state, consisting of position and yaw, $\xi = (x, y, z, \psi)^T$. The vehicle is allowed a maximum translational speed $v_{\max}$ and a constrained yaw rate $\dot{\psi}_{\max}$, both of which are assumed to be small, such that roll and pitch can be assumed to be zero. For slow maneuvering it suffices to use straight lines as tracking reference for the vehicle, specifically $\sigma_{k-1}^{k} = s\xi_k + (s-1)\xi_{k-1}$, with $s \in [0, 1]$. The connection cost of the reference path is considered to be the Euclidean distance, $c(\sigma_{k-1}^{k}) = \sqrt{(x_k - x_{k-1})^2 + (y_k - y_{k-1})^2 + (z_k - z_{k-1})^2}$. The maximum speed and the yaw rate limit are enforced when sampling the path to generate a reference trajectory.

Algorithm 1 Exploration Planner - Iterative Step

$\xi_0 \leftarrow$ current vehicle configuration
Initialize $\mathbb{T}$ with ξ_0 and, unless first planner call, also previous best branch
$g_{best} \leftarrow 0$ ▷ Set best gain to zero
$n_{best} \leftarrow n_0(\xi_0)$ ▷ Set best node to root
$N_{\mathbb{T}} \leftarrow$ Number of initial nodes in $\mathbb{T}$
while $N_{\mathbb{T}} < N_{\max}$ **or** $g_{best} = 0$ **do**
    Incrementally build $\mathbb{T}$ by adding $n_{new}(\xi_{new})$
    $N_{\mathbb{T}} \leftarrow N_{\mathbb{T}} + 1$
    if $\mathbf{Gain}(n_{new}) > g_{best}$ **then**
        $n_{best} \leftarrow n_{new}$
        $g_{best} \leftarrow \mathbf{Gain}(n_{new})$
    if $N_{\mathbb{T}} > N_{TOL}$ **then**
        Terminate exploration
$\sigma \leftarrow \mathbf{ExtractBestPathSegment}(n_{best})$
Delete $\mathbb{T}$
return σ

The employed depth camera is assumed to be mounted with a fixed pitch, to have a limited FoV described by vertical and a horizontal opening angle $[\alpha_v, \alpha_h]$, as well as a constraint on the maximum distance that it can sense. While the sensor limitation is $d_{\max}^{\text{sensor}}$, the algorithm uses a value $d_{\max}^{\text{planner}} \leq d_{\max}^{\text{sensor}}$ to determine the gain of a configuration. A lower $d_{\max}^{\text{planner}}$ ensures both robustness against suboptimal sensing conditions, as well as improved computational performance.

The depth information is incorporated in a probabilistic occupancy map, the specific implementation of which employs the octomap package [11] and provides further functionality for the gain computation and the collision checking of subspaces. Spaces of the shape of a box around the paths are checked to be free in the occupancy map.

B. Computational Complexity

As the planner should run onboard a mobile robot like an MAV or similar, short computation times and good scaling properties are crucial. The main scenario–dependent parameters that are relevant for these properties are the volume to be explored V and the resolution of the occupancy map r. In addition, the number of nodes in the tree $N_{\mathbb{T}}$ and the choice of the sensor range $d_{\max}^{\text{planner}}$ determine the duration of the path computation. For occupancy map queries in an octree [11] gives a logarithmic complexity in the number of voxels N_{voxel}, corresponding to $\mathcal{O}(\log(V/r^3))$. The construction of an RRT tree in a fixed environment is, as shown in [13], of complexity $\mathcal{O}(N_{\mathbb{T}} \log(N_{\mathbb{T}}))$, while the query for the best node in an RRT tree only scales with $\mathcal{O}(N_{\mathbb{T}})$. The number of voxels in the fixed volume around an edge to test for collision scales with $1/r^3$ and the complexity to check the $N_{\mathbb{T}} - 1$ edges in the occupancy map is $\mathcal{O}(N_{\mathbb{T}}/r^3 \log(V/r^3))$. Furthermore, for every node the gain has to be computed, considering a volume that is proportional to $V_{\text{sensor}} \propto (d_{\max}^{\text{planner}})^3$. The number of voxels to test is therefore $\approx \frac{V_{\text{sensor}}}{r^3}$. A ray cast to check visibility is performed for every voxel. Its complexity scales with the number of voxels on the ray $\mathcal{O}(d_{\max}^{\text{planner}}/r)$, resulting in $\mathcal{O}(d_{\max}^{\text{planner}}/r \log(V/r^3))$ for the visibility check and $\mathcal{O}((d_{\max}^{\text{planner}}/r)^4 \log(V/r^3))$ for one gain computation. As the construction of the RRT tree dominates its query, the total complexity of a single replanning step results in the sum of tree construction, collision–checking and gain computation

$$\mathcal{O}(N_{\mathbb{T}} \log(N_{\mathbb{T}}) + N_{\mathbb{T}}/r^3 \log(V/r^3) + N_{\mathbb{T}}(d_{\max}^{\text{planner}}/r)^4 \log(V/r^3))$$

Notably, the complexity only depends logarithmically on the volume of the exploration problem when the map resolution and planning horizon are fixed. For very large scenarios however, it may happen that $N_{\mathbb{T}}$ exceeds $N_{\max}$ in some iterations, especially when the majority of space has been explored already.

TABLE I: Apartment Exploration Scenario Parameters

Parameter	Value	Parameter	Value
Area	20x10x3m	Map resolution r	0.4m
$v_{\max}$	0.2m/s	$\dot{\psi}_{\max}$	0.75rad/s
FoV	$[60, 90]^\circ$	Mounting pitch	15°
$d_{\max}^{\text{planner}}$	2m	$d_{\max}^{\text{sensor}}$	5m
λ	0.5	RRT max edge length	1m
$N_{\max}$	15	Collision box	0.5x0.5x0.3m

V. Simulation Based Evaluation

In order to systematically evaluate the potential of the proposed exploration planner, simulation studies have been performed using a hexacopter MAV. Two scenarios of different size are considered, while in both the proposed algorithm is compared to a frontier–based approach (similar to [18]).

A. Simulation Environment

Since the planner works in a closed loop with the robot's perception, estimation and control loops, a detailed and realistic simulation is required. The Gazebo–based simulation environment RotorS [25] has been used along with the provided model of the AscTec Firefly hexacopter MAV [26]. It employs a stereo camera that gives real time depth images of the simulated environment. Its specifications are a FoV of $[60, 90]^\circ$ in vertical and horizontal direction, respectively, while it is mounted with a downward pitch angle of 15°. For all simulations, the size of the box model for collision detection is assumed to have a size of 0.5x0.5x0.3m.

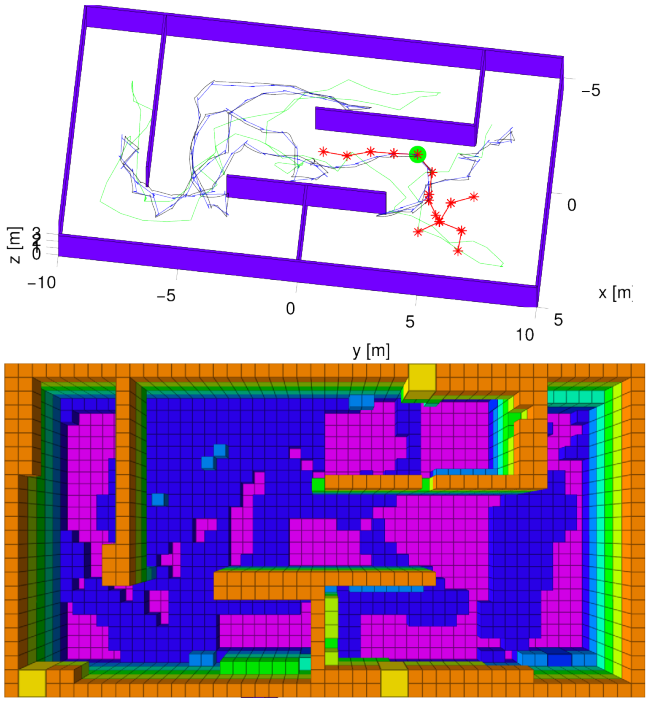

Fig. 3: The apartment setup is shown together with an instant of the exploration mission in the upper part. Blue is the reference path of the whole mission, while the vehicle response is depicted in black. The green point denotes the root of the RRT tree shown in red. An alternative exploration path computed by the frontier–based algorithm is shown as a green line. The floor and the roof are not visualized. In the lower part, the occupancy map is depicted, with the voxels colored according to their height.

B. Apartment Exploration Scenario

The first scenario refers to a 20x10x3m apartment space, divided in different rooms by walls as depicted in Figure 3. The vehicle starts in the center and navigates through narrow passages in order to explore the whole space. The resolution of the online built map is $r = 0.4$m and $N_{\max} = 15$ initial RRT iterations are performed. To ensure high accuracy, the maximum sensing range for the employed depth sensor is $d_{\max}^{\text{sensor}} = 5$m, while the planner only considers a maximum range of $d_{\max}^{\text{planner}} = 2$m. The parameter λ for Equation 1 is set to 0.5. Table I summarizes all employed parameters.

In order to assess the performance and overall quality of the proposed planner, a comparison with a frontier–based planner has been conducted ([18] was adapted to work with the occupancy map and the specific problem setup). For this purpose, free voxels in the neighbourhood of unmapped voxels have been considered as frontier voxels. They were checked to allow collision–free placement of the vehicle at their center and were evaluated for a set of randomly sampled orientations according to $\mathbf{Gain}(n) = \mathbf{Visible}(\mathcal{M}, \xi)e^{-\lambda l}$, where l is the distance determined by an RRT.

For both planners the scenario has been run 10 times, as the outcome is stochastic due to the use of the RRT algorithm. For the analysis, the total exploration time t_{tot} is split into execution time t_{ex} and computation time t_{comp}. The proposed planner displayed a mean total time for the exploration of $t_{tot}^{mean} = 501.9$s with a standard deviation of $t_{tot}^{\sigma} = 79.4$s, where the computation time was $t_{comp}^{mean} = 15.2$s with standard deviation $t_{comp}^{\sigma} = 2.7$s. A single iteration's computation lasted on average 0.153s. The histogram in Figure 4 shows, that only in very few replanning steps significantly more computation time was required. These were the cases where N_{max} iterations in the RRT tree were not enough to find a nonzero gain. The frontier algorithm consumed on average $t_{tot}^{mean} = 469.7$s, but also had more voxels that could not be inspected (V_{res}) at the time of termination. This is due to the limited choice of viewpoints only at the frontiers. When comparing the exploration progress over the execution time in Figure 4, even in this simple scenario the proposed planner performs slightly better. Furthermore, the total computation time for the frontier–based case is with an average of $t_{comp}^{mean} = 83.8$s on a significantly higher level. Because in every iteration the whole computed path is executed, t_{comp}^{mean} distributes on less iterations, resulting in an average replanning time of 5.7s, in which the robot waits for the next path segment to be computed. Sample results for paths of both planners are depicted in Figure 3, together with the occupancy map acquired by a mission using the proposed planner.

This simulation scenario showed that the performance of the proposed planner is not inferior to a frontier–based approach, despite the low number of viewpoints considered at a time. It even succeeds in finding some voxels that

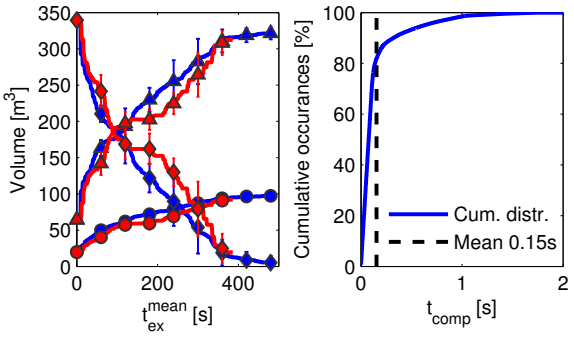

Fig. 4: On the left, statistics compare the receding horizon next–best–view planner (blue) with a frontier–based approach (red) for the apartment scenario. The depicted curves show the mean exploration progress over the execution time (computation time has been subtracted) for **occupied** o, **free** △ and **unmapped** ◊ volumes. The standard deviation is given every minute of execution time. On the right, a histogram shows the distribution of durations for the individual recomputation steps. While the average is 0.153s, significantly longer computation times ($N_{\mathbb{T}} > N_{\max}$) only occur very few times.

TABLE II: Bridge Exploration Scenario Parameters

Parameter	Value	Parameter	Value
Area	50x26x14m	Map resolution r	0.25m
$v_{\max}$	0.5m/s	$\dot{\psi}_{\max}$	0.75rad/s
FoV	$[60, 90]^\circ$	Mounting pitch	15°
$d_{\max}^{\text{planner}}$	2m	$d_{\max}^{\text{sensor}}$	10m
λ	0.2	RRT max edge length	3m
$N_{\max}$	30	Collision box	0.5x0.5x0.3m

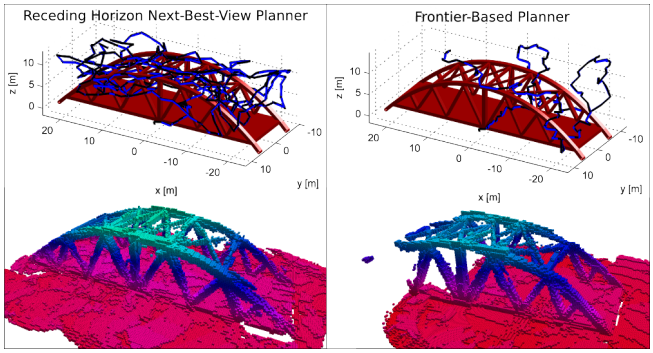

Fig. 5: The bridge model is displayed along with the computed path (blue), the simulated vehicle response (black) and the acquired occupancy map in the lower part of the figure. These results are given for the proposed planner on the left side and the compared frontier–based planner on the right. Note that the latter only made partial exploration due to excessive computation time for this large scale problem.

remain hidden to the compared algorithm. Moreover, its short computation time allows the seamless integration into the robot's control and path planning loops. While this very basic scenario highlights the ability of the presented planner to efficiently explore a given space, a more challenging scenario will reveal its high performance.

C. Exploration of a Space Containing a Bridge

A second simulation scenario refers to a 50x26x14m space containing a bridge [27], as depicted in Figure 5. Starting at the side of the bridge, the robot has to explore the whole space, while at the same time mapping the bridge in the occupancy map, the resolution of which is $r = 0.25$m. Due to the larger size, the higher resolution and the geometrical complexity of the bridge, this scenario is a much more challenging problem. $N_{\max} = 30$ RRT iterations are performed in every replanning. The sensor range is considered to be $d_{\max}^{\text{sensor}} = 10$m, while the range for the planning is $d_{\max}^{\text{planner}} = 2$m. Table II summarizes all employed parameters.

The total time for a single run has been $t_{tot} = 43.8$min, where in total $t_{comp} = 9.4$min have been spent on computation. A single replanning lasted on average only 1.6s. The median of only 1s reveals that a large portion of the computation time has been spent for the cases, where after $N_{\max}$ iterations no gain has been found. These replanning steps took up to 23s. In contrast, the frontier–based comparison run has been aborted after $t_{tot} = 1670.1$min. Until then, the planner spent $t_{comp} = 1660.4$min for computation and on average 25.9min per replanning step.

The progress over the total exploration time for both planners are displayed in Figure 6, while Figure 5 displays the paths on top of the simulated model, as well as the occupancy maps at the end of the executions.

This second, more complex scenario reveals the good scaling properties of the proposed planner, which is capable of handling large problems, without excessive runtime. At the same time, the implementation of an alternative frontier–based algorithm proves to be less suitable for more complex problems, as the computation time quickly grows infeasible.

VI. Experimental Evaluation

A real world experiment was conducted to further evaluate and demonstrate the performance of the proposed exploration strategy. Running with the limited onboard resources and in real–time, imposes fixed and tight limits on the amount of computations that are feasible.

A. Firefly Aerial Robot Platform

The experiment was performed using the AscTec Firefly hexacopter MAV [26] onboard of which the Visual–Inertial Sensor (VI–Sensor) developed by the Autonomous Systems Lab at ETH Zurich and Skybotix AG is integrated. The sensor provides stereo images tightly coupled with a high quality inertial sensor (IMU). A point cloud of the environment is constructed from the stereo image for the environment mapping, while a visual–inertial odometry algorithm is used for pose estimation [28], [29]. Using the full state estimate feedback, a trajectory tracking model predictive controller is implemented on the aerial robot that considers the vehicle dynamics and provides attitude and thrust reference for the low–level controller running on the autopilot provided by the MAV manufacturer. An external motion capture system (VICON) was used only to monitor the vehicle ground–truth state and provide the option to intervene in case of failure using a programmed "safety pilot".

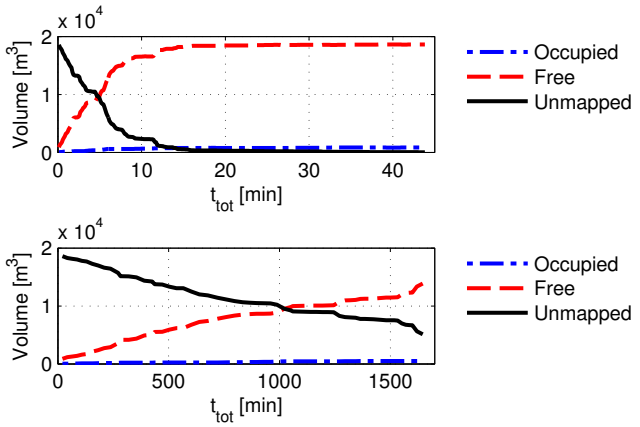

Fig. 6: These two figures depict the exploration progress for the proposed planner on the top and the frontier–based comparison algorithm on the bottom for the bridge scenario. The solid black line denotes the unmapped space, decaying as exploration progresses. The red dashed line denotes known free space and the blue alternately dashed and dotted line denotes occupied space.

Fig. 7: The AscTec Firefly hexacopter MAV is depicted, together with the scaffold structure used for the exploration experiment.

B. Indoor Exploration Scenario

The experimental scenario refers to a closed room with a size of 9x7x2m. It contains scaffold elements at the walls, which set high demands in terms of perception accuracy and robustness, and collision free navigation capability, as the structure consists of thin poles. The employed aerial robot and the scaffold structure are depicted in Figure 7. During the mission, the MAV motion is constrained by $v_{\max} = 0.25$m/s and $\dot{\psi}_{\max} = 0.3$rad/s, the translational and rotational speed limits, respectively. The attached stereo camera module's FoV is $[60, 90]^\circ$ and it is mounted with a downward pitch of 15°. For planning purposes, a sensing range of $d_{\max}^{\text{planner}} = 1$m is considered, while the sensor's range is set to $d_{\max}^{\text{sensor}} = 5$m. The λ-parameter for the gain computation is set to 0.5, which corresponds to a strong penalty on distance. Collision checking is performed in the $r = 0.2$m resolution occupancy map. All employed parameters are summarized in Table III, while Figure 8 shows the progress at distinct instants of the mission execution. The exploration starts at $t_{tot} = 0$, after an initialization motion to allow the computation of a first collision free path segment. The MAV subsequently

TABLE III: Scaffold Exploration Experiment Parameters

Parameter	Value	Parameter	Value
Area	9x7x2m	Map resolution r	0.2m
$v_{\max}$	0.25m/s	$\dot{\psi}_{\max}$	0.3rad/s
FoV	$[60, 90]^\circ$	Mounting pitch	15°
$d_{\max}^{\text{planner}}$	1m	$d_{\max}^{\text{sensor}}$	5m
λ	0.5	RRT max edge length	1.5m
$N_{\max}$	20	Collision box	1.2x1.2x0.5m

explores the space, first mostly by yawing, which is favored by the high λ-parameter and then also by moving to different regions of the space to explore. The exploration finishes at $t_{tot} = 253.37$s and after 58 planning iterations. Of the total time, $t_{comp} = 11.5$s have been used for computation, which corresponds to an average value of 0.199s per computation step (standard deviation 0.125s).

Overall, this experiment demonstrates the applicability of the proposed algorithm to exploration using real small aerial robots. Further the very short computation times highlight the high performance in computing the next path segment.

VII. Summary & Conclusion

Within this work, an exploration path planner was proposed that is capable of exploring a previously unknown area, constructing an occupancy map of the perceived environment. While collisions are avoided, good exploration paths are computed online, considering the updated model of the environment. Compared to a frontier–based planner, the proposed method offers improved scaling properties with respect to the size of the scenario, enabling the exploration of large areas. These properties have been demonstrated in simulation test cases, as well as in a challenging real world experiment, using a hexacopter rotorcraft, equipped with a stereo camera system. The implementation of the proposed planner is released as an open source code package for further development by the community [14].

References

[1] J. Nikolic, M. Burri, J. Rehder, S. Leutenegger, C. Huerzeler, and R. Siegwart, "A uav system for inspection of industrial facilities," in *Aerospace Conference, 2013 IEEE*. IEEE, 2013, pp. 1–8.

[2] S. Omari, P. Gohl, M. Burri, M. Achtelik, and R. Siegwart, "Visual industrial inspection using aerial robots," in *Applied Robotics for the Power Industry (CARPI), 2014 3rd International Conference on*. IEEE, 2014, pp. 1–5.

[3] R. Khanna, M. Möller, J. Pfeifer, F. Liebisch, A. Walter and R. Siegwart, "Beyond point clouds - 3d mapping and field parameter measurements using uavs," in *Emerging Technologies and Factory Automation (ETFA), 2015 IEEE International Conference on*, September 2015, (accepted).

[4] F. Colas, S. Mahesh, F. Pomerleau, M. Liu, and R. Siegwart, "3d path planning and execution for search and rescue ground robots," in *Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on*. IEEE, 2013, pp. 722–727.

[5] E. Galceran, R. Campos, N. Palomeras, D. Ribas, M. Carreras, and P. Ridao, "Coverage path planning with real-time replanning and surface reconstruction for inspection of three-dimensional underwater structures using autonomous underwater vehicles," *Journal of Field Robotics*, 2014.

[6] J. Rosenblatt, S. Williams, and H. Durrant-Whyte, "A behavior-based architecture for autonomous underwater exploration," *Information Sciences*, vol. 145, no. 1, pp. 69–87, 2002.

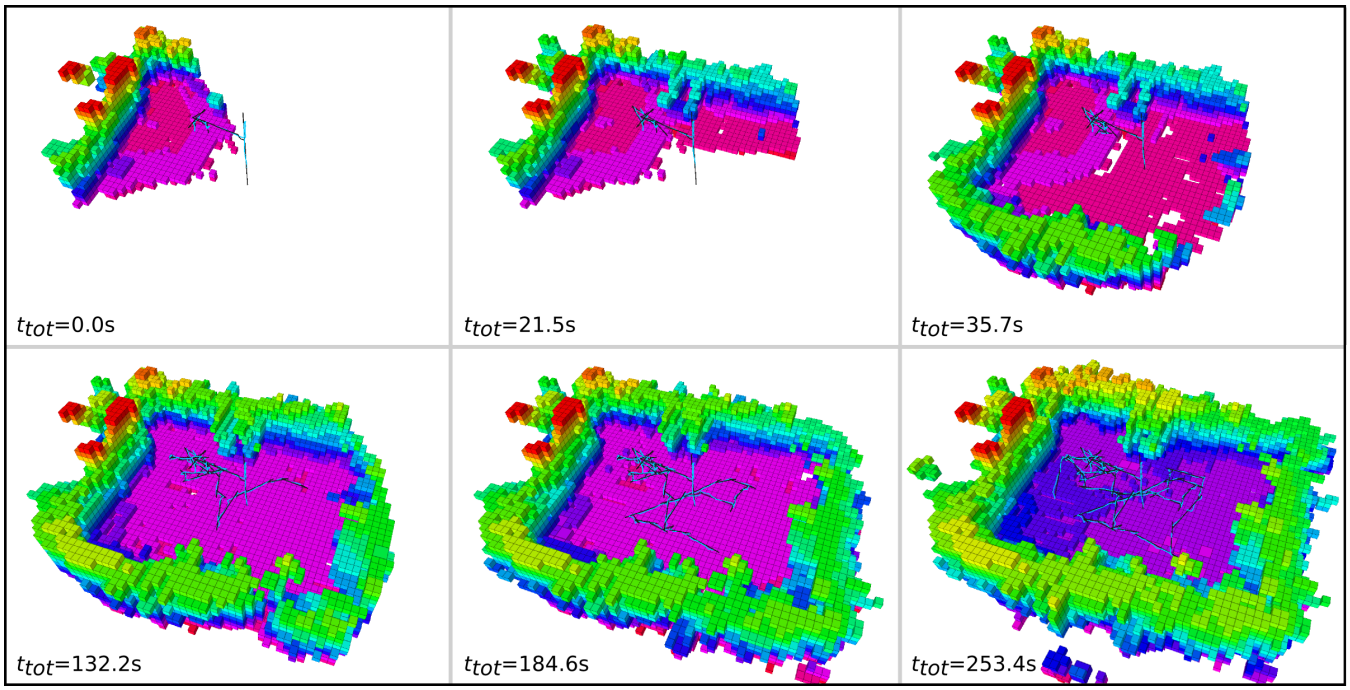

Fig. 8: The exploration experiment in a closed room is depicted. The colored voxels represent occupied parts of the occupancy map (colored according to height) while the computed path is given in black and the vehicle response in light blue. The initial phase of the exploration mission is dominated by yawing motions to maximize exploration without traveling large distances. Subsequently the MAV explores regions further away, to eventually accomplish its mission.

[7] A. Bircher, K. Alexis, M. Burri, P. Oettershagen, S. Omari, T. Mantel and R. Siegwart, "Structural inspection path planning via iterative viewpoint resampling with application to aerial robotics," in *Robotics and Automation (ICRA), 2015 IEEE International Conference on*, May 2015, pp. 6423–6430.

[8] F. S. Hover, R. M. Eustice, A. Kim, B. Englot, H. Johannsson, M. Kaess, and J. J. Leonard, "Advanced perception, navigation and planning for autonomous in-water ship hull inspection," *The International Journal of Robotics Research*, vol. 31, no. 12, 2012.

[9] J. I. Vasquez-Gomez, L. E. Sucar, R. Murrieta-Cid, and E. Lopez-Damian, "Volumetric next best view planning for 3d object reconstruction with positioning error," *Int J Adv Robot Syst*, vol. 11, p. 159, 2014.

[10] H. Surmann, A. Nüchter, and J. Hertzberg, "An autonomous mobile robot with a 3d laser range finder for 3d exploration and digitalization of indoor environments," *Robotics and Autonomous Systems*, vol. 45, no. 3, pp. 181–198, 2003.

[11] A. Hornung, K. M. Wurm, M. Bennewitz, C. Stachniss, and W. Burgard, "Octomap: An efficient probabilistic 3d mapping framework based on octrees," *Autonomous Robots*, vol. 34, no. 3, pp. 189–206, 2013.

[12] S. M. LaValle, "Rapidly-exploring random trees a new tool for path planning," 1998.

[13] 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.

[14] Andreas Bircher, Kostas Alexis, "Receding Horizon Next Best View Planner Code Release." [Online]. Available: https://github.com/ethz-asl/nbvplanner

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

[16] B. Kuipers and Y.-T. Byun, "A robot exploration and mapping strategy based on a semantic hierarchy of spatial representations," *Robotics and autonomous systems*, vol. 8, no. 1, pp. 47–63, 1991.

[17] 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.

[18] H. H. González-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.

[19] B. Adler, J. Xiao, and J. Zhang, "Autonomous exploration of urban environments using unmanned aerial vehicles," *Journal of Field Robotics*, vol. 31, no. 6, pp. 912–939, 2014.

[20] L. Heng, A. Gotovos, A. Krause, and M. Pollefeys, "Efficient visual exploration and coverage with a micro aerial vehicle in unknown environments," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2015.

[21] W. Burgard, M. Moors, D. Fox, R. Simmons, and S. Thrun, "Collaborative multi-robot exploration," in *Robotics and Automation, 2000. Proceedings. ICRA'00. IEEE International Conference on*, vol. 1. IEEE, 2000, pp. 476–481.

[22] A. Howard, L. E. Parker, and G. S. Sukhatme, "Experiments with a large heterogeneous mobile robot team: Exploration, mapping, deployment and detection," *The International Journal of Robotics Research*, vol. 25, no. 5-6, pp. 431–447, 2006.

[23] E. F. Camacho and C. Bordons, *Model Predictive Control*. Springer, 2003.

[24] K. Alexis, G. Nikolakopoulos, and A. Tzes, "Model predictive quadrotor control: attitude, altitude and position experimental studies," *Control Theory & Applications, IET*, vol. 6, no. 12, pp. 1812–1827, 2012.

[25] RotorS: An MAV gazebo simulator, "https://github.com/ethz-asl/rotors_simulator."

[26] Ascending Technologies GmbH, "http://www.asctec.de/."

[27] Bridge Model, 3D Warehouse, "https://3dwarehouse.sketchup.com/."

[28] M. Bloesch, S. Omari, M. Hutter, and R. Siegwart, "Robust visual inertial odometry using a direct ekf-based approach," in *Intelligent Robots and Systems (IROS), 2015 IEEE/RSJ International Conference on*. IEEE, 2015, p. to appear.

[29] 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.