sensors

MDPI

Article

Optimal Frontier-Based Autonomous Exploration in Unconstructed Environment Using RGB-D Sensor

Liang Lu *, Carlos Redondo and Pascual Campoy

Computer Vision and Aerial Robotics Group (CVAR), Centre for Automation and Robotics (C.A.R.), Universidad Politécnica de Madrid (UPM-CSIC), Calle José Gutiérrez Abascal 2, 28006 Madrid, Spain; carlos.redondop@alumnos.upm.es (C.R.); pascual.campoy@upm.es (P.C.)

* Correspondence: liang.lu@upm.es

Received: 15 October 2020; Accepted: 12 November 2020; Published: 14 November 2020

Abstract: Aerial robots are widely used in search and rescue applications because of their small size and high maneuvering. However, designing an autonomous exploration algorithm is still a challenging and open task, because of the limited payload and computing resources on board UAVs. This paper presents an autonomous exploration algorithm for the aerial robots that shows several improvements for being used in the search and rescue tasks. First of all, an RGB-D sensor is used to receive information from the environment and the OctoMap divides the environment into obstacles, free and unknown spaces. Then, a clustering algorithm is used to filter the frontiers extracted from the OctoMap, and an information gain based cost function is applied to choose the optimal frontier. At last, the feasible path is given by A* path planner and a safe corridor generation algorithm. The proposed algorithm has been tested and compared with baseline algorithms in three different environments with the map resolutions of 0.2 m, and 0.3 m. The experimental results show that the proposed algorithm has a shorter exploration path and can save more exploration time when compared with the state of the art. The algorithm has also been validated in the real flight experiments.

Keywords: aerial robot; autonomous exploration; information gain; RGB-D sensor

1. Introduction

During the last decade, the research focus on Micro Aerial Vehicles (MAVs) for search and rescue tasks has considerably increased [1–5]. Because of the small size and high maneuvering, MAVs can perform search and rescue missions more efficiently. Considering the payload of MAVs, only limited sensors can be carried on-board. For this reason, choosing the type of sensors and building an efficient system for MAVs is still challenging in order to perform such autonomous missions.

In the application of the MAVs, a large number of works have been carried out in the area of autonomous exploration [6–20]. MAVs-based autonomous exploration needs miniaturized sensors which can provide rich information to successfully fulfill the endeavor. Particularly, LIDARs or cameras are the main sensors used by the research or industrial society which are able to provide high dimensional information for the autonomous exploration or inspection tasks. Compared with LIDARs, cameras such as RGB cameras, RGB-D cameras or event cameras have a huge advantage in the price and size.

In this paper, an RGB-D camera which can provide point clouds is used as the main sensor to perform the perception tasks. Then, the information from the RGB-D camera is used for the mapping and planning module to find a feasible path that has more environment information. The OctoMap for mapping and Information Gain (IG) is used to search the optimal goal position and orientation for exploration. With the optimal goal, a safe corridor-based A* algorithm is used to plan safety and feasible path. Finally, position control is applied to fly the path.

There are three main innovations in our work:

First of all, the efficent strategy for generation frontier points, filtering frontier points using the k-means++ and finding the optimal frontier using IG information. Secondly, combining a safe corridor-based A* planning algorithm with a position controller to move to the optimal frontier. At last, the algorithm is integrated with our aerial robot framework and tested in a custom edge computing aerial platform.

The remainder of the paper is organized as follow. Section 2 presents the related works. The robot architecture and environment definition are described, and the problem is formulated in Section 3. The proposed optimal frontiers generation and motion planning algorithm are explained in Section 4. Section 5 introduces the algorithm complexity. The experimental results and discussion are shown in Sections 6 and 7, respectively, and finally, Section 8 concludes the paper and summarizes future research directions.

2. Related Works

A number of methods have been proposed during the last few decades addressing the problem of autonomous exploration in unconstructed environments. There are two fundamental types of approaches: the frontier-based approaches and sampling-based approaches. The frontier-based approaches try to maximize the exploration efficiency by selecting and exploring the optimal frontiers between the known and unknown area of a map, while the sampling-based approaches randomly generate robot states and search the path which can maximize the information gathered in the environment. There are also works [8,9] that combine the frontier-based and sampling-based approaches. In recent years, the learning-based strategies [10,11,21] such as reinforcement learning, are also used to solve the autonomous exploration problem. In this paper, the classic autonomous navigation algorithms are mainly discussed. In the remainder of the section, firstly the literature using the frontier-based approach is introduced and then the work using the sampling-based method are explained; finally, the research using the hybrid exploration strategy is summarized.

2.1. Frontier-Based Autonomous Exploration

The frontier-based exploration approach was first introduced by B. Yamauchi in the year of 1997 [12]. Considering the limited computing resources on-board MAV, S. Shen et al. [22] proposed an autonomous exploration algorithm only using the information of the known occupied space in the current map. After evaluating the information in the regions determined by the evolution of a stochastic differential equation, the ones of most significant particle expansion correlated to unexplored space and then MAV started to explore these unknown regions. In 2013, C. Dornhege and A. Kleiner proposed a frontier-void-based autonomous exploration approach [23]. They combined two concepts of voids, which are the unexplored volumes and frontiers. Their approach has been evaluated in a mobile platform with a five DOF manipulator. Recently, S. Bai et al. [13] used Bayesian optimization to predict mutual information (MI) of the frontiers and then trained a Gaussian process (GP) to estimate the MI in the robot's action space and chose the next exploration step. C. Wang et al. [14] proposed a multi-objective reward function that could minimize both the map entropy and the path cost during the exploration process, then the frontier that had the best reward would be chosen as the goal, finally, a potential field is construed to guide the robot to the goal. H. Umari et al. [15] presented a local and global RRT-based frontier detection for candidate frontiers generation and then a function with information gain and navigation cost is applied to select the optimal frontier. Because of the local and global RRT-based frontier generation, this approach has a high algorithm complexity. T. Cieslewski et al. [16] proposed a frontier selection method for high speed flight. They used a function to select the frontier that had the minimum change in the velocity to reach it. Their strategy could significantly reduce the exploration time because of the high exploration speed.

The have been more works in the last two years, an effective exploration strategy using expected information gain is proposed by E. Palazzolo and C. Stachniss [24]. Their method could reduce the risk of collision, while still maximizing the information gain in the exploration process. C. Gomez et al. introduced a topological frontier-based exploration using semantic information [25]. They combined frontier-based concepts with behavior-based strategies in order to build a topological representation of the environment.

2.2. Sampling-Based Autonomous Exploration

As one of the most famous sampling-based exploration methods, Receding Horizon "Next-Best-View" Planner (RHC-NBVP) [17] was firstly proposed by A. Bircher et al. RHC-NBVP repeatedly expanded a rapidly-exploring random tree and selected the best next node to explore using an objective function that considered the set of visible and unmapped voxels from robot configuration. This approach had two main disadvantages: (1) keep repeatedly expanding RRT is computing expensive; (2) select the optimal next step in the field of view might lead to a sub-optimal solution. C. Papachristos et al. [18] presented an RHC-NBVP considering about the localization and mapping uncertainty. This implementation helped the planner run in challenging environments like dark or clutter environments. A two stage optimized next-view planning framework is developed by Z. Meng et al. [26]. The two-stage planner consists of a frontier-based boundary coverage planner and a planner returns an exploration path with the consideration of global optimality in the context of accumulated space information. Thinking about reducing the computing resources of the RHC-NBVP, C. Witting et al. proposed the history-aware strategy [19]. Their planner would maintain and use a history of visited places so that they can improve the sampling efficiency. The Visual Saliency–aware NBVP is introduced by T. Dang et al. [20]. They used visual attention to build the maps that are annotated regarding the visual importance and saliency of different objects and entities in the environment. Then the visual saliency information was used by the planner to find the next best step. In the year of 2020, L. Schmid et al. [27] presented an RRT*-inspired online informative path planning algorithm. This algorithm could achieve global coverage and maximize the utility of a path in a global context, using a single objective function.

By comparing with the algorithms above, a frontier-based exploration algorithm using global information gain that could avoid the sub-optimal path is used. In order to achieve good exploration, the robot that performs the exploration missions is moving at a low velocity.

3. Preliminary

In this section, firstly the architecture of the robot and the definition of the environment are given and then the problem is formulated.

3.1. Robot Architecture and Environment Definition

The robot used in this paper is an aerial robot that has six degrees of freedom. These six degrees of freedom are the translational movements in the X, Y, and Z axes and the rotational movements which are roll, pitch, and yaw. The onboard sensor of the robot is a visual–inertial sensor with an RGB-D camera that can be used to perform localization and realize the environment.

A 3D occupancy grid map generated by using OctoMap [28] is applied to describe the environment and the gathered data from the RGB-D camera are used to provide the point clouds for updating the map.

The 3D occupancy map is organized by voxels, and every voxel has one state which is free, occupied or unknown. With OctoMap, every voxel has an occupied possibility. The value of the occupied possibility is in the range of 0 to 1. For the voxel x, the entropy $e(x)$ can be calculated using the occupied possibility $p(x)$ as in Equation (1).

$$e(x) = -(p(x))log(p(x)) - (1 - p(x))log(1 - p(x)) \tag{1}$$

The entropy can be used to describe the uncertainty of the voxel. Lower values of the entropy imply a lower value of the uncertainty. The sum of the voxels' entropy can be used to guide the autonomous exploration to reduce the map uncertainty.

In this work, the voxels whose occupied possibility value are higher than 0.5 are thought as the obstacle voxels.

3.2. Problem Formulation

The goal of the autonomous exploration is to explore an unknown environment as fast as possible. To improve the efficiency of the exploration algorithm, the width, length, and height of the environment are previously defined in this work.

The width, length and height of the environment are assumed w, l and h, the volume of every voxel is V_{voxel}, the guide goal pose is $G_i(x_i, y_i, z_i, \theta_i)$ and the start pose of the robot is $\{P_0(x_0, y_0, z_0, \theta_0) | -w/2 < x_0 < w/2, -l/2 < y_0 < l/2, 0 < z_0 < h\}$. The robot starts to move from P_0 to G_i. The exploration process ends when the algorithm cannot find a new G_i. If the sum of V_{voxel} is equal to $w \times l \times h$, it is thought of as a successful running of the exploration algorithm. The exploration process can be seen in Figure 1.

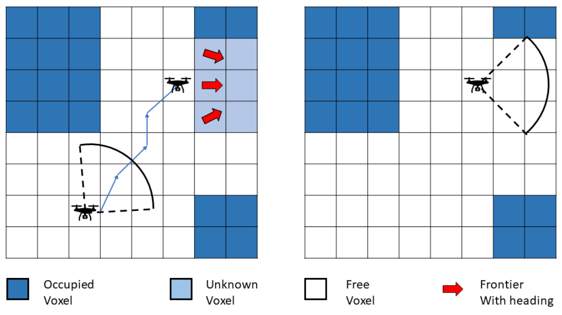


Figure 1. The schematic diagram for the exploration process. The robot starts exploration by moving to the selected frontier and finishes exploration when there is no new frontier found.

4. Methodology

In this section, the optimal frontier selection approach and the motion planning and control strategy are explained in detail. Firstly the optimal frontiers generation approach, which generates the goal for the motion planning module, is described and then how we move the robot to the goal using the motion planning and control module is explained.

4.1. Optimal Frontiers Generation

The points between known and unknown area in the occupancy map are selected as frontier points. Then, the candidate points are generated around these frontier points. The candidate points are generated using the formula below.

$$\begin{aligned}
&R_{i+1} = R_i + \Delta R;\\
&\theta_{i+1} = \theta_i + \Delta\theta;\\
&i \in 0, 1, \dots, i, \dots, n-1, n;\\
&R_{min} \le R_i \le R_{max};\\
&0 \le \theta_i \le 2\pi;
\end{aligned} \tag{2}$$

$$\begin{aligned}
&(X_c, Y_c, Z_c, \theta_c)\\
&= (X_f + R_i \times cos(\theta_i), Y_f + R_i \times sin(\theta_i), Z_f, \theta_i);\\
&i \in 0, 1, \dots, i, \dots, n-1, n;
\end{aligned} \tag{3}$$

In Equation (2), θ_i is the yaw angle generated by the candidates generation algorithm. R_{min} and R_{max} is the minimum and maximum distances from the frontiers to the candidates. ΔR and $\Delta\theta$ are predefined increments for updating R_i and θ_i. In this article, ΔR and $\Delta\theta$ are chosen as 0.5 m and 12 °. In Equation (3), X_c, Y_c, Z_c and θ_c are the x, y, z coordinate and yaw angle of the candidates, respectively. X_f, Y_f and Z_f are the x, y, z coordinates of the frontiers.

The distance from the candidate point to the frontier points is set as D_{cf} and the distance to the robot is set as D_{cr}. With the candidates generated, firstly the candidates which are in the unknown area are removed and then the candidates whose D_{cf} and D_{cr} are smaller than a predefined value are removed.

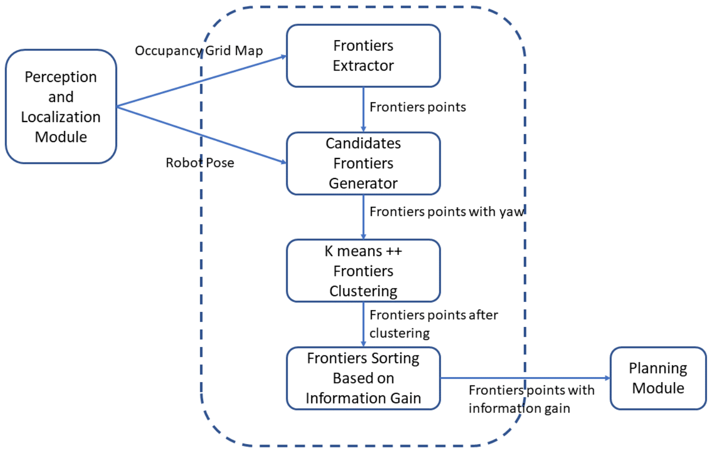
With the generated candidates, the Kmeans++ algorithm is used for candidates clustering. By using the Kmeans++ algorithm, the number of the candidates can be reduced which can speed up the optimal frontier selection process.

The information value of the candidates after clustering can be calculated by using the information gain cost function [29]. By using the information value, the candidates can be sorted. The information gain cost function can be seen from Equation (4).

$$G_v = \sum_{\forall r \in R_v} \sum_{\forall x \in X} e(x) \tag{4}$$

In Equation (4), G_v is the information gain of the candidate points on the view v. R_v is the set of rays cast through the candidate points on the view v in the robot's field of view. X is the set of voxels each ray traverses through. $e(x)$ is the mapping uncertainty of the voxel. We thought of the sum of $e(x)$ of the candidate points on the view v as the information gain value.

After the candidates are sorted, they are sent to the motion planning and control module, the whole process of the optimal frontiers generation is given by Figure 2.



Perception and Localization Module
Occupancy Grid Map
Robot Pose
Frontiers Extractor
Frontiers points
Candidates Frontiers Generator
Frontiers points with yaw
K means ++ Frontiers Clustering
Frontiers points after clustering
Frontiers Sorting Based on Information Gain
Frontiers points with information gain
Planning Module

Figure 2. The flow chart for the optimal frontier generation process.

4.2. Motion Planning and Control

With the frontiers points from the optimal frontiers generation module, the motion planning module firstly selected the frontier point, which has the smallest value as the goal point.

Firstly, the A* path planning algorithm is used to plan a primitive path from the current point to the goal point. Then, the safe corridors are generated based on the path point. The safe corridor generation approach is similar, as in [30]. The occupancy grid including the path point is inflated from each of its six faces. The inflation of each face finishes when it reaches the obstacle voxel or the size of the inflated voxel reaches a predefined value. Finally, a path is replanned through the center of every safe corridor. By using this strategy, the replanned path is farther away from the obstacle than the previous path. The process is shown in Figure 3.

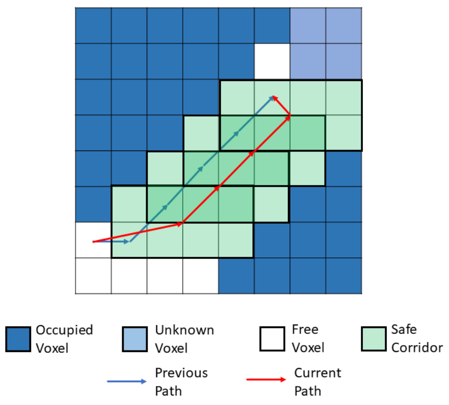


Figure 3. The schematic diagram for safe corridor generation process.

For the yaw planning, the yaw angle of the endpoint always looks towards the direction that has the largest IG gain while the other path points always look towards the next path point.

If the planner cannot find a feasible path to the goal or the planning time is higher than the predefined value, the current goal point are discarded and choose another frontier point with smaller value as the goal point.

After getting a feasible path, the planner sends the path points one by one to the motion controller. When there is a new obstacle detected in the path, the controller will stop the robot and let the planner find another feasible path. If there is no feasible path, the algorithm will generate candidates and search the feasible path again. The whole exploration process stops when the algorithm cannot find new frontiers. the whole process of the motion planning and control module is given by Figure 4.

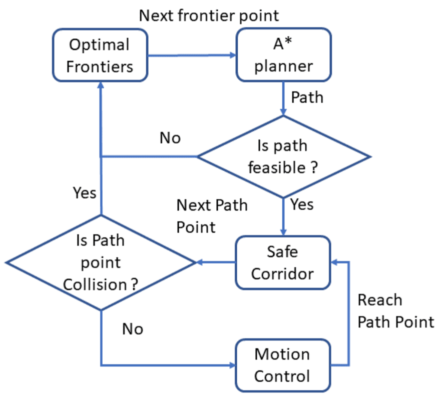


Figure 4. The flow chart for the motion planning and control process.

5. Algorithm Complexity

The algorithm complexity is important when the computing resources of the agent are limited. For every run of the algorithm, firstly the frontiers are generated, then the information gain of the frontiers is computed and the best frontier with the most information is selected, and finally the A* algorithm is used to search the feasible path.

1. *Frontiers Generation*: the complexity of the frontier generation algorithm depends on the raw frontiers generation and Kmean++ clustser algorithm. As can be seen from Algorithm 1, The complexities of raw frontier algorithm and Kmean++ clustering algorithm are $O(2\pi \times (R_{max} - R_{min})/(\Delta yaw \times \Delta R))$ and $O(3 \times i \times k \times m)$, respectively. i, k, and m are the iteration times of the Kmean++ algorithm, the number of frontiers defined and the number of frontiers generated by raw frontiers generation, respectively.

2. *Information Gain Computing*: The complexity of the gain estimation of each frontier point is the number of rays n in the FOV times the length of rays l to the obstacle divided by the resolution of the map r, which is $O(n \times l/r)$.
3. *Path Search*: when using A* algorithm searching the feasible path in a 3D occupancy map, we assume A* algorithm can search six directions when expanding a new node and the number of nodes in the path is a. The complexity of the A* search is $O(6 \times a)$. For the complexity of the safe corridor generation algorithm, it can be calculated by the minimum distance L_j to the obstacle in the $x, y, and\ z$ directions of every path node times the number of nodes a divided by the resolution of the map r, which is $O(a \times \sum_{j=0}^{a} L_j/r)$.
4. *Total Computational Complexity*: The total computational complexity per run of the algorithm is $O(2\pi \times (R_{max} - R_{min})/(\Delta yaw \times \Delta R) + 3 \times i \times k \times m + n \times l/r + a \times \sum_{j=0}^{a} L_j/r)$.

Algorithm 1 Autonomous Exploration Path Generation.

```
Input: $Map$, $P_{init}$
Output: $Path$
1: $\{F_0, F_1, ..., F_m\} \leftarrow ExtraFrontier(P_{init}, Map)$
2: $n \leftarrow 0$
3: for $n < m$ do
4:     $yaw \leftarrow 0$
5:     for $yaw < 2\pi$ do
6:         $R \leftarrow R_{min}$
7:         for $R < R_{max}$ do
8:             $F_n(x) \leftarrow F_n(x) - R\cos(yaw)$
9:             $F_n(y) \leftarrow F_n(y) - R\sin(yaw)$
10:            $F_n(z) \leftarrow F_n(z)$
11:            $F_n(yaw) \leftarrow yaw$
12:            if $F_n \in unknown$ then
13:                $F_{group}(n) \leftarrow F_n$
14:            end if
15:            $R \leftarrow R + \Delta R$
16:        end for
17:        $yaw \leftarrow yaw + \Delta yaw$
18:    end for
19:    $n \leftarrow n + 1$
20: end for
21: $\{F_0, F_1, ..., F_k\} \leftarrow KmeanCluster(F_{group}, k)$
22: $IG_{max} \leftarrow 0; n \leftarrow 0; opt \leftarrow 0$
23: for $n < k$ do
24:     $IG_{F_n} \leftarrow IG(F_n)$
25:     if $IG_{F_n} > IG_{max}$ then
26:         $IG_{max} \leftarrow IG_{F_n}$
27:         $opt \leftarrow n$
28:     end if
29:     $n \leftarrow n + 1$
30: end for
31: $Path \leftarrow AstarSearch(P_{init}, F_{opt})$
32: while $P_i \in Path$ do
33:     while $P_i$ is not in collision do
34:         $P_{new} \leftarrow Inflate(P_i(x), P_i(y), P_i(z))$
35:         if $P_i$ is in collision then
36:             Replace $P_i$ with $P_{new}$
37:             Break While
38:         end if
39:     end while
40: end while
```

The process of the whole exploration algorithm is shown in Algorithm 1. The autonomous exploration path generation algorithm can be divided into three parts. These three parts are frontiers generation parts (from line 1 to line 20), optimal frontiers selection part (from line 21 to line 30), corridor-based A* path planning part (from line 31 to line 40). The detailed descriptions about these three parts can be seen in the Methodology.

6. Experiments and Results

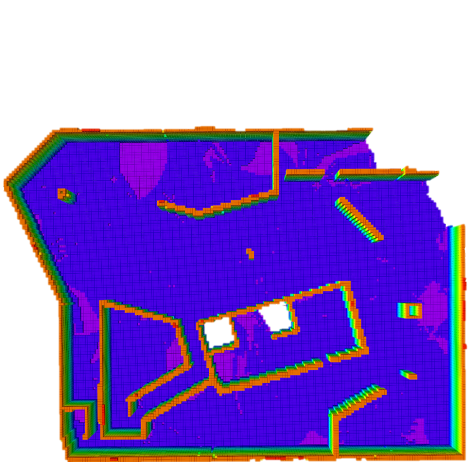
6.1. Experimental Setup

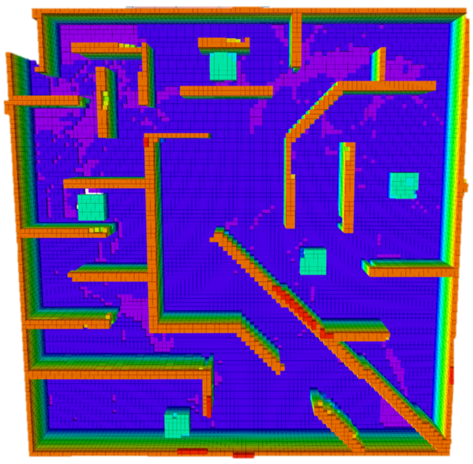
For the simulation experiments, the experiments using the Robot Operating System (ROS) [31] are run on top of Ubuntu 18.04. The Gazebo simulator is used to provide the environment model and the RotorS simulation [32] is used to provide the physical model parameters of robot. All the experiments run on a laptop with Intel Core i7-8750H at 2.2 GHz. The robot in the simulation is the AscTec firefly, which is equipped with a RGB-D camera with a horizontal field of view of 135° and vertical field of view of 60°. The sensing range of the RGB-D camera is set as 5 m. The camera has an angle of 6° looking down to the ground. In the simulation experiments, the ground truth is used to perform the robot localization and the robot starts in the origin with yaw angle of zero.

To evaluate our exploration algorithm, it is tested in three different environments. These environments are a small-size environment, a medium-size environment, and a big-size environment. They are named as the apartment environment, the maze environment, and the office environment. These three environments can be seen in Figure 5. The size of the flat environment is 15 m $\times$ 14 m $\times$ 3 m, the size of the maze environment is 20 m $\times$ 20 m $\times$ 2.5 m. The height of the office environment is 2.5 m and the area of it is 800 m^2. Our algorithm is compared with two different algorithms. One algorithm uses Bayesian optimization and the Gaussian process to predict the mutual information for frontier selection [13] and the other algorithm uses the Euclidean distance to choose the nearest frontier to explore. The main differences between our algorithm and these two algorithms are the kmeans++ for candidate clustering and using information gain for calculating the information value of the frontiers. The algorithms are also tested in the same environment with resolutions.

The platform for the experiment is that shown in Figure 6. As can be seen from the figure, the aerial platform is the Parrot Bebop 2, the on-board computer is the Jetson TX2 4GB module with orbit carrier and the RGB-D sensor is the Intel Realsense D435i, respectively. The depth camera D435i is part of the Intel RealSense D400 series of cameras, a lineup that takes Intel's latest depth-sensing hardware and software and packages them into easy-to-integrate products. Adding an IMU allows the application to refine its depth awareness in any situation where the camera moves. This opens the door for rudimentary SLAM and tracking applications allowing better point-cloud alignment. The minimum depth distance of the depth camera is 0.0105 m while the horizontal and vertical FOVs of are 86 and 57°, respectively. The whole syste, including the controller, mapping and exploration module is run on the Jetson TX2 with a Dual-core NVIDIA Denver 2 64-bit CPU and NVIDIA Pascal^{TM} Architecture GPU with 256 CUDA cores. The exploration algorithm is integrated with the Aerostack frame work [33] in the real flight. The system architecture used in this work is shown in Figure 7.

As can be seen in Figure 7, there are five main parts in the system architecture that are human operator, execution module, communication module, sensor system, and actuator system. For the part of the human operator, three basic commands are used. These three basic commands are *take off*, *position control* and *land*. The basic commands can be sent to the onboard computer via Wifi. The other four parts including the autonomous exploration algorithm run from the onboard computer and the outputs are the roll, pitch, yaw, and thrust commands. The control commands are sent to Parrot Bebop 2 via a USB cable that connect the onboard computer and drone.


(**a**) Office environment.


(**b**) Maze environment.

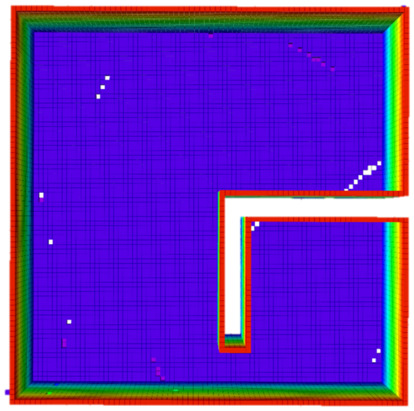

(**c**) Flat environment.

Figure 5. Three simulation environments for testing the exploration algorithm.

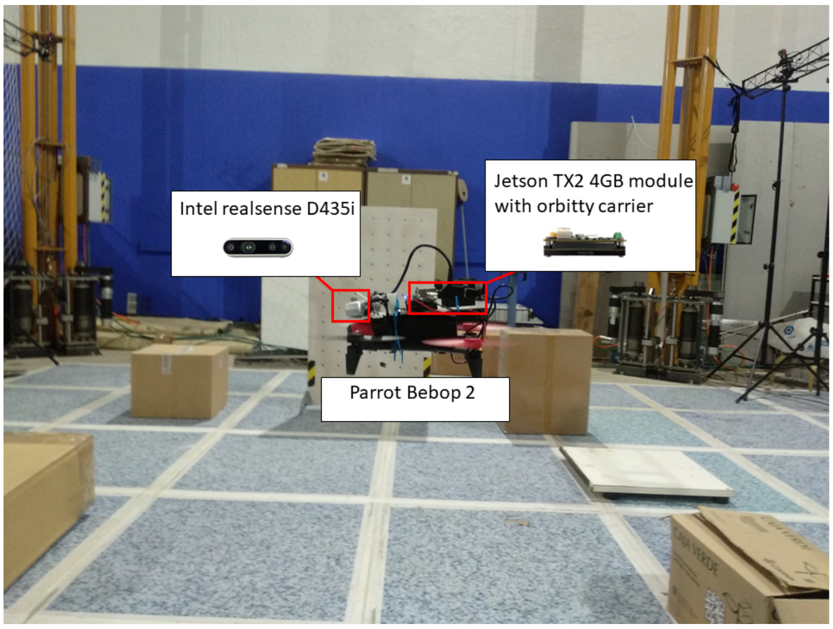


Figure 6. Aerial platform used in the real flight experiment.

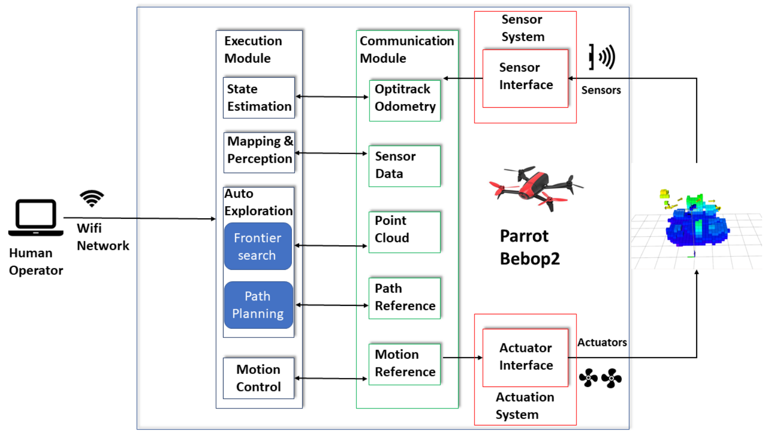
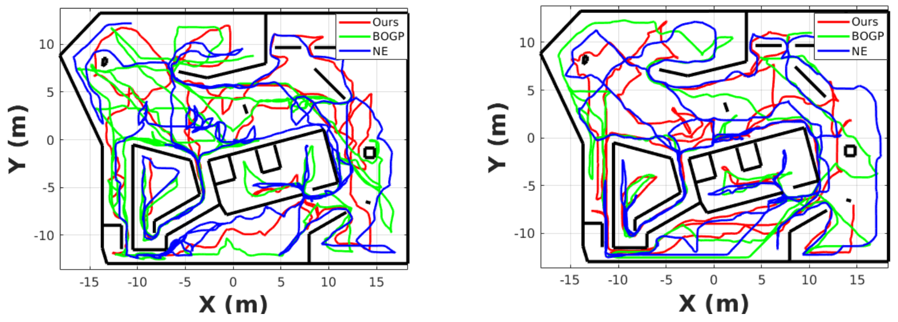


Figure 7. The system architecture used in the real flight experiment (based on the Aerostack framework). The optitrack odometry is provided by the optitrack motion capture system. The position and steering tracking error of it is less than 0.03 mm and 0.05°, respectively.

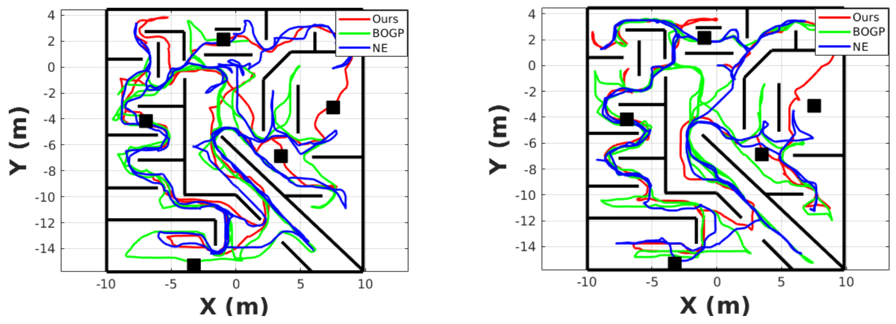
6.2. Simulation Experiments

The simulation results of the exploration path in the office, maze, and flat environments are shown in Figure 8. The start point of the exploration algorithms in the three environments with different mapping resolutions are all $(0, 0, 1)$ m.



(**a**) Office environment (res = 0.2). The path lengths of the proposed, NF and BOGP based algoritm are 310.82 m, 371.50 m, 296.32 m, respectively.

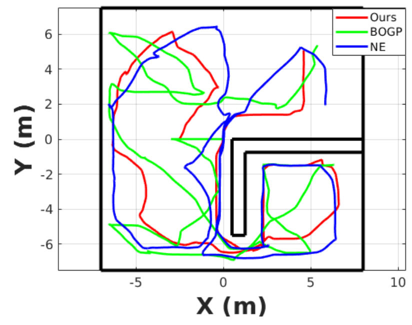
(**b**) Office environment (res = 0.3). The path lengths of the proposed, NF and BOGP based algoritm are 381.19 m, 460.78 m, 333.92 m, respectively.



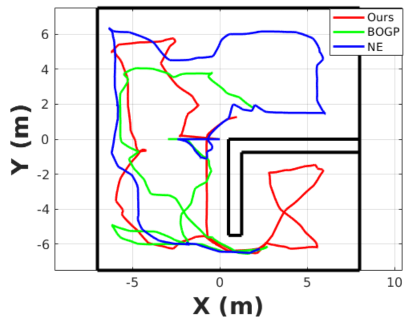
(**c**) Maze environment (res = 0.2). The path lengths of the proposed, NF and BOGP based algoritm are 163.72 m, 257.65 m, 230.18 m, respectively.

(**d**) Maze environment (res = 0.3). The path lengths of the proposed, NF and BOGP based algoritm are 154.44 m, 219.33 m, 311.46 m, respectively.

Figure 8. *Cont.*



(**e**) Flat environment (res = 0.2). The path lengths of the proposed, NF and BOGP based algoritm are 65.12 m, 90.73 m, 87.87 m, respectively.



(**f**) Flat environment (res = 0.3). The path lengths of the proposed, NF and BOGP based algoritm are 62.32 m, 51.13 m, 48.96 m, respectively.

Figure 8. The simulation results of the exploration path of one run. The red line , green line, and blue line are our algorithm, the algorithm using Bayesian optimization and Gaussian process and the algorithm using the nearest frontier, respectively.

We limit the maximum exploration time of the office, maze, and flat environments as 1500 s, 1200 s, and 500 s. In this article, BOGP and NF are used to express the Bayesian optimization and the Gaussian process-based algorithm and the nearest frontier-based exploration algorithm. Each algorithm is run five times in the flat, maze and office environment with different resolutions. Table 1 shows the exploration time of these three algorithm under different map resolutions. The results of BOGP-based algorithm in the office environment with the map resolution of 0.2 m are not included because the exploration time is far more than the limited time.

Table 1. Comparison of the proposed algorithm and baseline algorithm. The bold number in the table means the best performance in the comparison of each group in the table.

Environment	Map Resolution (m)	Exploration Algorithm	Exploration Time (s)	Path Length (m)
flat env	0.2	*Ours*	**255 ± 9.12**	**75.56 ± 14.76**
		NF	326 ± 31.21	89.37 ± 1.93
		BOGP	411 ± 23.33	92.43 ± 6.46
	0.3	*Ours*	**209 ± 12.72**	63.16 ± 1.19
		NF	227 ± 43.84	67.15 ± 8.27
		BOGP	217 ± 23.33	**52.48 ± 4.97**
maze env	0.2	*Ours*	**808 ± 72.12**	**192.26 ± 40.36**
		NF	905 ± 46.67	271.72 ± 19.90
		BOGP	1082 ± 98.99	228.34 ± 2.60
	0.3	*Ours*	**650 ± 59.3**	**180.27 ± 35.27**
		NF	761 ± 23.33	223.11 ± 19.50
		BOGP	1051 ± 151.32	306.18 ± 7.47
office env	0.2	*Ours*	**1495 ± 75.57**	**355.58 ± 30.29**
		NF	1609 ± 141.02	366.50 ± 6.41
		BOGP	-	-
	0.3	*Ours*	**1361 ± 81.63**	379.50 ± 7.42
		NF	1515 ± 113.61	449.06 ± 10.48
		BOGP	1576 ± 207.01	**374.41 ± 57.26**

6.3. Real Flight Experiments

In order to validate the proposed algorithm, a real flight experiment is given that runs the algorithm on-board an MAV. The experiment was conducted in a 4 m × 2 m × 2 m scenario equipped with a Optitrack motion capture system providing the MAV pose. Figure 9 shows the scenario for the MAV to navigate and the mapping for visualization. As can be seen from the figure, the cylider is put as an obstacle and the MAV will navigate around it. The cylinder obstacle is put at the position of $(1.3, -0.1)/\text{m}$. The height and radius of the cylinder are 1.26 m and 0.1 m, respectively. The boxes and board show the boundary of the scenario. The exploration time and the length of the exploration path is 44.9 s and 9.0398 m. *P*1 and *P*2 is the selected optimal frontier points. As can be seen from Figure 10, the robot first flew to *P*1 and then to *P*2. When the robot arrived *P*2, the final optimal frontier point *P*3 was selected. The exploration process was finished when robot arrived *P*3.

A video description about the experiments is shown at the following link: https://vimeo.com/455285058/f969e81451.

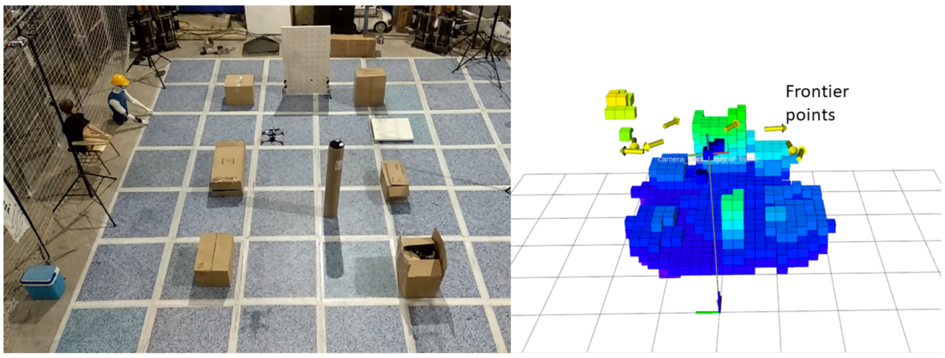


Figure 9. (**Left**) A photograph of the MAV and the scenario. (**Right**) Map of the scenario the experiment was conducted in. Voxels are coloured based on their height and the objects have been cropped for visualization.

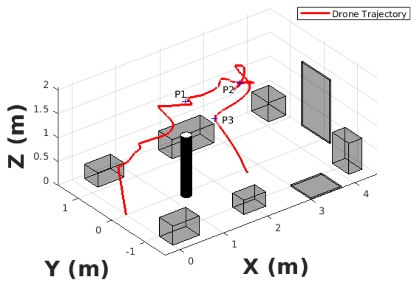
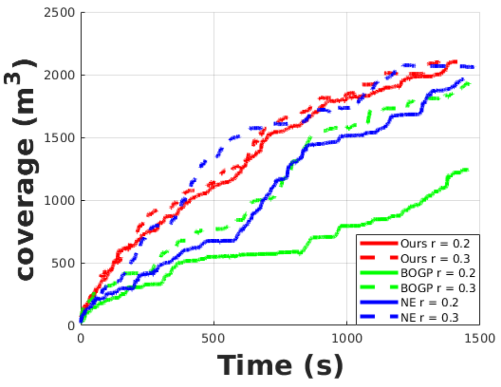


Figure 10. The results of the real flight experiments. The red line is the trajectory of the aerial robot.

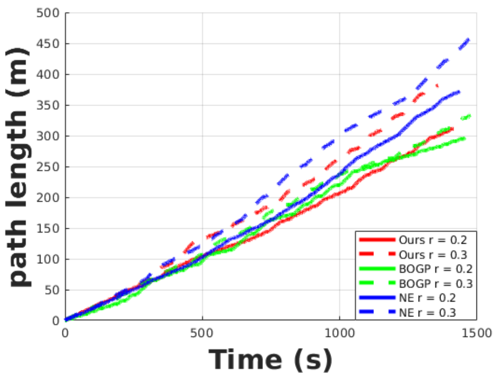
7. Discussion

As can be seen from Figure 8, if the exploration time is not limited, all the exploration algorithms can explore the whole environment when the map resolution is 0.2 m and 0.3 m.

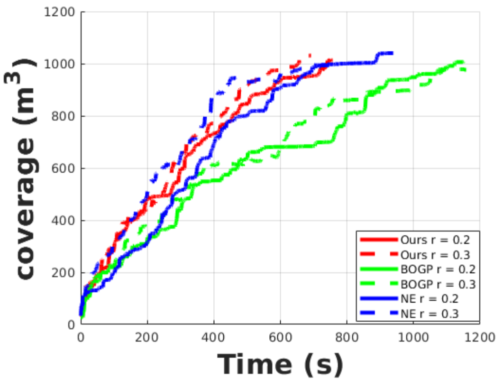
The performance improvement of the proposed algorithm can be more clear when the exploration time is limited. As shown in Figure 11a,c,e, when the resolution of the map is 0.2 m (the solid line in the figure), the proposed algorithm (red line) can finish the exploration tasks in the limited time in all the scenarios. For BOGP based and NF based algorithm, it can finish the exploration tasks but takes a longer time than the proposed algorithm. In Figure 11a, the BOGP-based exploration cannot finish the exploration in the limited time when the map resolution is 0.2 m (the coverage of the whole scenario is 2000 m^3, while it only reaches around 1250 m^3). It is because the Bayesian optimization and Gaussian processes take a long time to analyze the map and this becomes more clear when performing in a big environment like the office scenario. As can be seen from Table 1, when the map resolution is 0.2 m, the proposed algorithm takes 255 s, 808 s, and 1495 s on average to finish exploration, while BOGP-based and NF-based algorithms take a much longer time. When the map resolution is increased to 0.3 m, as shown in Figure 11a,c,e, and Table 1, the performance of NF based algorithm is close to the proposed algorithm but the proposed algorithm still can finish the task faster.



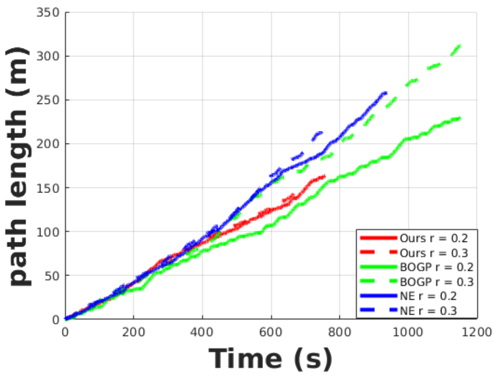
(**a**) The change in the voxel coverage in the office environment.



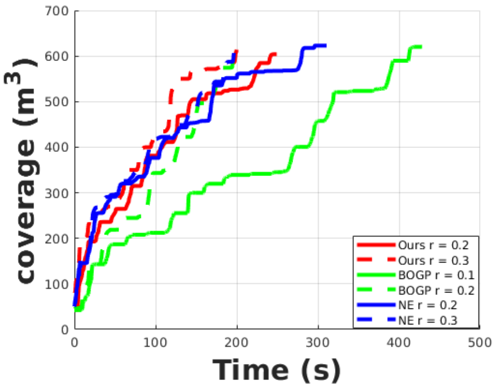
(**b**) The change oin the exploration path length in the office environment.



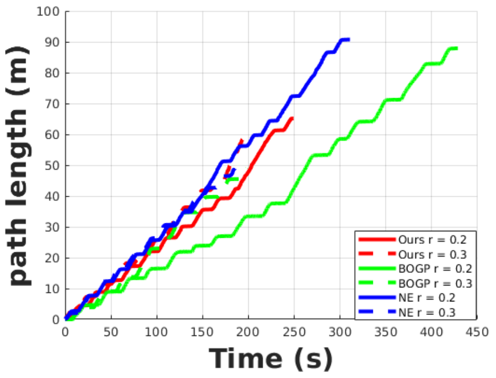
(**c**) The change in the voxel coverage in the maze environment.



(**d**) The change in the exploration path length in the maze environment.



(**e**) The change in the voxel coverage in the flat environment.



(**f**) The change in the exploration path length in the flat environment.

Figure 11. One run of the changing of the voxel coverage and exploration path in the office, maze, and flat environment. the red line, green line, and blue line are the results of the algorithm using Bayesian optimization and gaussian process and the algorithm using the nearest frontier, respectively. The solid line and the dashed line mean the resolution of mapping is 0.2 m and 0.3 m.

Figure 11b,d,f show the length of the exploration path changing during the exploration process. It can be seen in Figure 11d,f and Table 1 that the proposed algorithm has a shorter exploration path than the NF-based algorithm in all the three scenarios and has a shorter path than the BOGP-based algorithm in the maze scenario with both resolutions and flat scenario with the resolution of 0.2 m.

The results show that both optimal frontier selecting functions of the proposed and BOGP-based algorithm can help the exploration algorithm follow a shorter path for fulfilling the exploration task. In the office environment, with a resolution of 0.2 m, our algorithm has a longer exploration path length because the BOGP-based method only explores around half of the whole environment in the limited time.

Figure 11 shows one run for each method with different resolutions. Repeated experiments confirm that these curves are representative.

8. Conclusions and Future Works

In this article, an autonomous exploration strategy using the information gain to select the optimal frontier is presented. Firstly, an RGB-D camera to sense the environment and OctoMap is used to express the obstacle, free, and unknown space in the environment. Secondly, the Kmean++ algorithm is used to filter the frontiers and an information gain cost function is applied to select the optimal frontier to explore. Finally, A* path planner is applied to find the path and a safe corridor generation algorithm is used to move the path point far away from the obstacle before sending the path to the controller. The proposed algorithm has been compared with BOGP- and NF-based algorithms in three different environments with different map resolutions, in which the experimental results show that the proposed algorithm can save more exploration time than the other two algorithms and it also has a shorter exploration path.

The future work will focus on reducing the algorithm complexity and achieve higher exploration efficiency. A reinforcement learning-based exploration approach will also be considered in the future. The proposed exploration algorithm will also be implemented with Visual Inertial Odometry to perfrom the large environment exploration.

Author Contributions: Conceptualization, L.L.; methodology, L.L.; software, L.L.; validation, L.L. and C.R.; formal analysis, L.L.; investigation, L.L.; resources, L.L.; data curation, L.L.; writing—original draft preparation, L.L.; writing—review and editing, L.L., C.R. and P.C.; visualization, L.L.; supervision, P.C.; project administration, P.C.; funding acquisition, L.L. and P.C. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the Spanish Ministry of Science, Innovation and Universities (COMCISE RTI2018-100847-B-C21, MCIU/AEI/FEDER, UE) and the Chinese Scholarship Council (CSC) for the Scholarship of the first author.

Acknowledgments: The work reported in this paper is sponsored by the Chinese Scholarship Council (CSC). We also thank Alejandro Rodriguez-Ramos, Hriday Bavle and Adrian Carrio for their sincere advice and help during the real flight experiments.

Conflicts of Interest: The authors declare no conflict of interest.

Abbreviations

The following abbreviations are used in this manuscript:

MAVs	Micro Aerial Vehicles
IG	Information Gain
MI	Mutual Information
GP	Gaussian Process
RRT	Rapid Random Tree
RHC-NBVP	Receding Horizon “Next-Best-View” Planner
FOV	Field of View
ROS	Robot Operating System
BOGP	Bayesian Optimization and Gaussian Process
NF	Nearest Frontier

References

1. Fu, C.; Ye, J.; Xu, J.; He, Y.; Lin, F. Disruptor-Aware Interval-Based Response Inconsistency for Correlation Filters in Real-Time Aerial Tracking. *IEEE Trans. Geosci. Remote Sens.* **2020**. [CrossRef]
2. Fu, C.; Lin, F.; Li, Y.; Chen, G. Correlation Filter-Based Visual Tracking for UAV with Online Multi-Feature Learning. *Remote Sens.* **2019**, 11, 549. [CrossRef]
3. Carrio, A.; Sampedro, C.; Rodriguez-Ramos, A.; Campoy, P. A review of deep learning methods and applications for unmanned aerial vehicles. *J. Sens.* **2017**, 2017, 3296874. [CrossRef]
4. Bejiga, M.B.; Zeggada, A.; Nouffidj, A.; Melgani, F. A Convolutional Neural Network Approach for Assisting Avalanche Search and Rescue Operations with UAV Imagery. *Remote Sens.* **2017**, *10*, 100. [CrossRef]
5. Sampedro, C.; Bavle, H.; Rodriguez-Ramos, A.; Carrio, A.; Suarez-Fernandez, R.A.; Sanchez-Lopez, J.L.; Campoy, P. A fully-autonomous aerial robotic solution for the 2016 International Micro Air Vehicle competition. In Proceedings of the 2017 International Conference in Unmanned Aircraft Systems (ICUAS), Miami, FL, USA, 13–16 June 2017; pp. 989–998.
6. McGuire, K.N.; De Wagter, C.; Tuyls, K.; Kappen, H.J.; De Croon, G.C.H.E. Minimal navigation solution for a swarm of tiny flying robots to explore an unknown environment. *Sci. Robot.* **2019**, *4*, 1–14. [CrossRef] [PubMed]
7. Pérez-Higueras, N.; Jardón, A.; Rodríguez, Á.; Balaguer, C. 3D Exploration and Navigation with Optimal-RRT Planners for Ground Robots in Indoor Incidents. *Sensors* **2020**, *20*, 220. [CrossRef] [PubMed]
8. Selin, M.; Tiger, M.; Duberg, D.; Heintz, F.; Jensfelt, P. Efficient Autonomous Exploration Planning of Large-Scale 3-D Environments. *IEEE Robot. Autom. Lett. (RA-L)* **2019**, *4*, 1699–1706. [CrossRef]
9. Dai, A.; Papatheodorou, S.; Funk, N.; Tzoumanikas, D.; Leutenegger, S. Fast Frontier-based Information-driven Autonomous Exploration with an MAV. *arXiv* **2020**, arXiv:2002.04440.
10. Niroui, F.; Zhang, K.; Kashino, Z.; Nejat, G. Deep Reinforcement Learning Robot for Search and Rescue Applications: Exploration in Unknown Cluttered Environments. *IEEE Robot. Autom. Lett. (RA-L)* **2019**, *4*, 610–617. [CrossRef]
11. Chen, T.; Gupta, S.; Gupta, A. Learning Exploration Policies for Navigation. In Proceedings of the International Conference on Learning Representation(ICLR), New Orleans, LO, USA, 6–9 May 2019.
12. Yamauchi, B. A Frontier-Based Approach for Autonomous Exploration. In Proceedings of the 1997 IEEE International Symposium on Computational Intelligence in Robotics and Automation (CIRA), Monterey, CA, USA, 10–11 July 1997; pp. 146–151.
13. Bai, S.; Wang, J.; Chen, F.; Englot, B. Information-theoretic exploration with Bayesian optimization. In Proceedings of the 2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Daejeon, Korea, 9–14 October 2016; pp. 1816–1822.
14. Wang, C.; Meng, L.; Li, T.; De Silva, C.W.; Meng, M.Q. Towards autonomous exploration with information potential field in 3D environments. In Proceedings of the 2017 18th International Conference on Advanced Robotics (ICAR), Hong Kong, China, 10–12 July 2017; pp. 340–345.
15. Umari, H.; Mukhopadhyay, S. Autonomous robotic exploration based on multiple rapidly-exploring randomized trees. In Proceedings of the 2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Vancouver, BC, Canada, 24–28 September 2017; pp. 1396–1420.
16. Cieslewski, T.; Kaufmann, E.; Scaramuzza, D. Rapid exploration with multi-rotors: A frontier selection method for high speed flight. In Proceedings of the 2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Vancouver, BC, Canada, 24–28 September 2017; pp. 2135–2142.
17. Bircher, A.; Kamel, M.; Alexis, K.; Oleynikova, H.; Siegwart, R. Receding Horizon "Next-Best-View" Planner for 3D Exploration. In Proceedings of the 2016 IEEE International Conference on Robotics and Automation (ICRA), Stockholm, Sweden, 16–21 May 2016; pp. 1462–1468.
18. Papachristos, C.; Khattak, S.; Alexis, K. Uncertainty-aware receding horizon exploration and mapping using aerial robots. In Proceedings of the 2017 IEEE International Conference on Robotics and Automation (ICRA), Singapore, 29 May–3 June 2017; pp. 4568–4575.
19. Witting, C.; Fehr, M.; Bähnemann, R.; Oleynikova, H.; Siegwart, R. History-Aware Autonomous Exploration in Confined Environments Using MAVs. In Proceedings of the 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Madrid, Spain, 1–5 October 2018; pp. 1–9.

20. Dang, T.; Papachristos, C.; Alexis, K. Visual Saliency-Aware Receding Horizon Autonomous Exploration with Application to Aerial Robotics. In Proceedings of the 2018 IEEE International Conference on Robotics and Automation (ICRA), Brisbane, QLD, Australia, 21–25 May 2018; pp. 2526–2533.
21. Ramezani Dooraki, A.; Lee, D.-J. An End-to-End Deep Reinforcement Learning-Based Intelligent Agent Capable of Autonomous Exploration in Unknown Environments. *Sensors* **2018**, *18*, 3575. [CrossRef] [PubMed]
22. Shen, S.; Michael, N.; Kumarm, V. Autonomous indoor 3D exploration with a micro-aerial vehicle. In Proceedings of the 2012 IEEE International Conference on Robotics and Automation, Saint Paul, MN, USA, 14–18 May 2012; pp. 9–15.
23. Dornhege, C.; Kleiner, A. A frontier-void-based approach for autonomous exploration in 3D. *Adv. Robot.* **2013**, *27*, 459–468. [CrossRef]
24. Palazzolo, E.; Stachniss, C. Effective Exploration for MAVs Based on the Expected Information Gain. *Drones* **2018**, *2*, 9. [CrossRef]
25. Gomez, C.; Hernandez, A.C.; Barber, R. Topological Frontier-Based Exploration and Map-Building Using Semantic Information. *Sensors* **2019**, *19*, 4595. [CrossRef]
26. Meng, Z.; Qin, H.; Chen, Z.; Chen, X.; Sun, H.; Lin, F.; Ang, M.H. A Two-Stage Optimized Next-View Planning Framework for 3-D Unknown Environment Exploration, and Structural Reconstruction. *IEEE Robot. Autom. Lett.* **2017**, *2*, 1680–1687.
27. Schmid, L.; Pantic, M.; Khanna, R.; Ott, L.; Siegwart, R.; Nieto, J. An Efficient Sampling-Based Method for Online Informative Path Planning in Unknown Environments. *IEEE Robot. Autom. Lett. (RA-L)* **2020**, *5*, 1500–1507. [CrossRef]
28. Hornung, A.; Wurm, K.M.; Bennewitz, M.; Stachniss, C.; Burgard, W. OctoMap: An efficient probabilistic 3D mapping framework based on octrees. *Auton. Robot.* **2013**, *34*, 189–206. [CrossRef]
29. Delmerico, J.; Isler, S.; Sabzevari, R.; Scaramuzza, D. A comparison of volumetric information gain metrics for active 3D object reconstruction. *Auton. Robot.* **2018**, *42*, 197–208. [CrossRef]
30. Chen, J.; Liu, T.; Shen S. Online generation of collision-free trajectories for quadrotor flight in unknown cluttered environments. In Proceedings of the 2016 IEEE International Conference on Robotics and Automation (ICRA), Stockholm, Sweden, 16–21 May 2016; pp. 1476–1483.
31. Quigley, M.; Gerkey, B.; Conley, K.; Faust, J.; Foote, T.; Leibs, J.; Berger, E.; Wheeler, R.; Ng, A. ROS: An open-source Robot Operating System. In Proceedings of the Open-source Software Workshop of 2009 International Conference on Robotics and Automation (ICRA), Kobe, Japan, 12–17 May 2016; pp. 1476–1483.
32. Furrer, F.; Burri, M.; Achtelik, M.; Siegwart, R. *RotorS—A Modular Gazebo MAV Simulator Framework*; Robot Operating System (ROS); Studies in Computational Intelligence; Koubaa, A., Ed.; Springer: Cham, Switzerland, 2016; pp. 595–625.
33. Sanchez-Lopez, J.L.; Molina, M.; Bavle, H.; Pérez, C.S.; Fernandez, R.A.S.; Campoy, P. A Multi-Layered Component-Based Approach for the Development of Aerial Robotic Systems: The Aerostack Framework. *J. Intell. Robot. Syst.* **2017**, *88*, 683–709. [CrossRef]