Efficient and High Path Quality Autonomous Exploration and Trajectory Planning of UAV in an Unknown Environment

Leyang Zhao 1, Li Yan 1,*, Xiao Hu 1, Jinbiao Yuan 2 and Zhenbao Liu 2

1 School of Geodesy and Geomatics, Wuhan University, Wuhan 430079, China; leyangzhao@whu.edu.cn (L.Z.); huxiao@whu.edu.cn (X.H.)
2 School of Civil Aviation, Northwestern Polytechnical University, Xi’an 710068, China; yuanjinbiao0414@126.com (J.Y.); liuzhenbao@nwpu.edu.cn (Z.L.)
* Correspondence: lyan@sgg.whu.edu.cn

Abstract: The ability of an autonomous Unmanned Aerial Vehicle (UAV) in an unknown environment is a prerequisite for its execution of complex tasks and is the main research direction in related fields. The autonomous navigation of UAVs in unknown environments requires solving the problem of autonomous exploration of the surrounding environment and path planning, which determines whether the drones can complete mission-based flights safely and efficiently. Existing UAV autonomous flight systems hardly perform well in terms of efficient exploration and flight trajectory quality. This paper establishes an integrated solution for autonomous exploration and path planning. In terms of autonomous exploration, frontier-based and sampling-based exploration strategies are integrated to achieve fast and effective exploration performance. In the study of path planning in complex environments, an advanced Rapidly Exploring Random Tree (RRT) algorithm combining the adaptive weights and dynamic step size is proposed, which effectively solves the problem of balancing flight time and trajectory quality. Then, this paper uses the Hermite difference polynomial to optimize the trajectory generated by the RRT algorithm. We named proposed UAV autonomous flight system as Frontier and Sampling-based Exploration and Advanced RRT Planner system (FSEPlanner). Simulation performs in both apartment and maze environment, and results show that the proposed FSEPlanner algorithm achieves greatly improved time consumption and path distances, and the smoothed path is more in line with the actual flight needs of a UAV.

Keywords: unmanned aerial vehicles; autonomous exploration; path planning; octrees; frontier-based exploration; rapidly exploring random tree

1. Introduction

Unmanned aerial vehicles (UAVs) have good maneuverability and a strong hovering ability [1]. They are an ideal platform for surveillance, search and rescue in narrow indoor and outdoor environments [2]. Micro-UAVs equipped with airborne sensors are an ideal platform for autonomous navigation in complex and narrow environments and can solve problems such as exploration, mapping, search and rescue [3,4]. Navigating a micro-UAV in a chaotic and unknown environment is a challenging problem. To perform these tasks effectively, UAVs must not only explore the unknown environment around them and detect obstacles but also plan and execute a collision-free, dynamic and feasible trajectory [5]. Ideally, UAVs should not rely on external positioning systems such as GNSS, and all exploration, mapping, and planning processes should run on the UAVs themselves [6].

The comprehensive exploration of an unknown environment was the earliest problem of motion planning, was gradually combined with SLAM, and evolved into a comprehensive problem of autonomous motion systems including planning, navigation, and control [7,8]. Since the exploration algorithm run on the UAV board, it should be fast and lightweight. Traditionally, autonomous exploration is divided into two basic methods: frontier-based methods and sampling-based methods. A frontier-based method attempts
to maximize map coverage by identifying and exploring the frontier between known and unknown parts of a map, while a sampling-based method have also been employed where the objective is to sample the “next–best–view” [9,10].

After the exploration of an unknown environment by a UAV is addressed, path planning can be carried out to accurately reach the target point. Traditionally, UAV path planning is roughly divided into two types according to the degree of acquisition of prior environmental information: global path planning based on complete environmental information and local path planning based on partial sensor information.

In this article, our goal is to use UAV to provide systematic solutions for the autonomous exploration and planning of unknown environments (see Figure 1). Frontier-based and sampling-based exploration are combined to sample the candidate next view in the map frontier. Trajectory planning is based on the dynamic rapidly exploring random tree (RRT), and an advanced RRT path planning algorithm with dynamic step size and adaptive weight is proposed. Finally, trajectory optimization can satisfy the dynamic constraints of the UAV. The main contributions of this strategy are as follows.

- It combines frontier-based exploration with sampling-based exploration. Via implicit voxel grouping in the octree graph representation, frontier voxels can be regarded as clusters, thereby avoiding the computationally expensive steps of frontier voxel clustering in original frontier-based methods.
- Original RRT algorithm can easily fall into a local minimal area. This paper introduced the dynamic step size and adaptive weight in UAV path planning system based on the rapid exploration tree. The purpose of planning the optimal trajectory in the task environment based on dynamic step size and adaptive weights, so as to improve search efficiency, increasing success rate and improving quality of generated paths.
- To avoid irrationality of the planned path, UAV dynamic constraints are introduced in the planning process to avoid the situation where the climbing angle and the turning angle are large. Hermite difference polynomial is used for smooth out the twisted sections of the original path.

Figure 1. The procedure of autonomous navigation of UAV equipped with vision sensors in an unknown environment.

The rest of the paper is organized as follows: Section 2 presents the related work of autonomous exploration and path planning. Section 3 describes the system overview. Sections 4 and 5 presents the proposed autonomous exploration and path planning method. Section 6 discusses the experimental results of the method. Finally, we conclude our work and future research directions in Section 7.

2. Related Work

2.1. Autonomous Exploration

Autonomous exploration is to find a set of sensor poses along the frontier of an unknown space to ultimately explore the entire space considering path costs (such as time, length, or energy) [11]. Autonomous exploration deliberately seeks out parts of the area in the unknown environment, restore the real environment to the greatest extent, and keep searching until the information is acceptable. The main task of mobile robot autonomous exploration is to determine the desired motion position of the robot in the next step, and finally obtain the most unknown and correct environment information with the shortest collision free path in the global range. Although there are various methods, the two most mature heuristic methods for UAV exploration are frontier-based methods and sampling-based information collection methods [12]. The frontier-based methods recognize the frontier between the known space and the unknown space in a map and
repeatedly select a frontier as the target, which eventually leads to a complete exploration.
Sampling-based methods usually sample the view configuration in a sub-optimal next-best view (NBV) manner. Table 1 shows the advantage and difficulties of each main used autonomous exploration methodologies.

Table 1. The main used autonomous exploration methodologies with each advantage and difficulties.

| Autonomous Exploration Algorithm | Advantage | Difficulties |
|----------------------------------|-----------|--------------|
| Frontier-based exploration [13] | Perform well in 2-dimension environment | Redundant frontiers generating due to environmental occlusion and noise |
| VFH + plus bug algorithm [14]   | Perform well in sparse environments | Require the planner knows the location of the goal and assume the robot has perfect positioning |
| NBV [15]                        | High exploration coverage | Random sampling not always detect unexplored areas fast |
| Bircher’s NBV [16]              | Minimize the uncertainty of robot positioning and tracking marks | High computational complexity and long exploration time. |
| Directly sample candidate NBV [17]| Proposed the term of safe region, higher in exploration coverage | High precision of relative positioning is required |

Yamauchi [13] defines the border between the known space and the unknown space as the frontier and realizes the robot’s exploration of the entire space by controlling the robot to move towards the newly discovered frontier. Heng [14] uses the vector field histogram + (VFH+) algorithm for local navigation and a frontier-based exploration algorithm to propose a plan for drones to fly autonomously in unknown environments. Moreover, they adopt the bug algorithm of autonomous wall tracking in sparse environments, with poor frontier exploration performance.

Some researchers use sampling-based methods for autonomous exploration [18]. The basic idea here is to sample viewpoints in the explored map that might facilitate target exploration. Because the configuration space is randomly sampled, these planners can also achieve different optimization goals without changing the basic motion planning algorithm. The concept of NBV was first proposed by Connolly [15]. The author’s goal is to obtain the complete model of a scene by calculating a series of coverage views. Bircher [19] uses the NBV in a 3D exploration algorithm, their method iterates between sampling the accessible viewpoints in the RRT and executing the most informative path. The detection algorithm based on the Receding horizon NBV scheme, which builds a random tree for rapid exploration (RRT) by sampling and viewing the configuration and accumulating the information gain of each branch. This method is higher in exploration coverage, although the exploration time is longer.

Papachristos extends Bircher’s NBV to address the uncertainty in positioning and mapping between viewpoints [16]. Gonzalez-Banos [17] propose a method to directly sample candidate views in the configuration space. They determine the NBV by evaluating the sample using a utility function.

2.2. Path Planning

In the past few decades, path planning for mobile robots has been an interest topic in robotics field. The terms of UAVs motion planning are challenging because these vehicles have fast, complex and uncertain dynamics and strict payload limitations; they are combined with real-time navigation issues in 3D space and involve continuous interaction with the environment [20]. The goal of UAV motion planning is to generate collision-free trajectories in real time for vehicle dynamic constraints to pass through a three-dimensional environment with obstacles [21]. Many algorithms designed to solve this problem rely
on a decomposition hierarchy method. First, a smooth constraint is applied to form a trajectory that conforms to the path, and a control loop is used to follow the trajectory to solve the path planning problem [22]. The advantage of this method is that any original kinematics/dynamics unconstrained algorithm in the field of robotics can be used in the path planning stage, and it is easy to integrate the motion planning algorithm with the existing low-level dynamic controller. Most researchers divide this problem into global and local path planning: (1) Global path planning is based on complete environmental a priori information and can also be called static or offline path planning. (2) Local path planning involves path planning in an unknown environment with real-time sensor information and can also be called dynamic or online path planning.

Path planning means that a moving body (usually a mobile robot) finds an optimal or sub-optimal route that can connect the starting state and the target state and does not overlap with obstacles according to performance indicators such as distance, time, and energy. This state space is usually represented by an occupancy grid or lattice to describe the position of an object in the environment. These planning techniques are summarized from two technical aspects: graph search-based and sampling-based planning algorithms. Table 2 shows the advantage and difficulties of each main used path planning methodologies.

Planner based on graph search algorithms such as A Star [23] find the optimal solution in the connectivity graph, while D Star [24] and Field D Star [25] search dynamic graphs to find the optimal solution. The use of graph search methods is limited by the discretization of the workspace and their performance degradation in high-dimensional spaces. Lattice-based work uses motion primitives to generate state lattices and combines them with graph search algorithms, but the discretization problem is still inevitable.

Sampling-based planners try to solve timing constraints, they carry out planning in a high-dimensional space. Deterministic methods cannot achieve this kind of planning. The configuration space or state space is randomly sampled to find the internal connectivity in the space. The most promising algorithms for UAVs are the probabilistic road map (PRM) method [26] and RRT [27]. Although the idea of connecting randomly sampled points from the state space is essential to the two methods, they differ in the construction of the graphs that connect these points. The PRM algorithm and its variants are multiple query methods. First, a collision-free roadmap is constructed, which contains a rich set of collision-free roadmaps. Then, the shortest path connecting the initial state and the final state is calculated to obtain the result [28]. Although multiple queries are valuable in highly structured environments, most online planning problems do not require multiple queries because robots move from one environment to another or the environment is not known a priori. Moreover, in some applications, computing roadmaps a priori may be computationally challenging or even infeasible. For these applications, a customized incremental sampling planning algorithm (such as the RRT) is more effective. The RRT performs a random search in the navigation area to achieve rapid planning in a semi-structured space while considering non-holonomic constraints. In particular, the RRT can be an effective foundation applicable to systems with differential constraints and nonlinear dynamics, as well as purely discrete or hybrid systems.

The RRT algorithm has been expanded in several directions, and many applications have been found in the robotics field and elsewhere. In contrast to the basic RRT algorithm, Karaman S. and Frazzoli E. of MIT proved the theory and proposed the RRT Star algorithm [29]. By constructing the cost function and using this as the criterion, they selected the node with the smallest cost in the neighborhood of the node to be expanded on the tree as the parent node. On this basis, the nodes on the current tree need to be reconnected after each iteration to ensure a low computational complexity and obtain an asymptotically optimal solution. RRT*FN [30] establish a fixed number of nodes, which randomly removes a leaf node in the tree in every iteration. It is no longer a single expansion from the initial position to the target area, which greatly improves the search efficiency and speed. As the number of sampling points increases, the probability of the RRT algorithm converging
to the optimal solution tends to zero. Informed RRT Star [31] improves the convergence speed of RRT Star by introducing a heuristic, similar to the way in which A Star improves upon Dijkstra’s algorithm. RRT-Connect [32] was proposed to reduce the exploration time of the exploration tree. The principle is to construct the growth tree from the initial point and the target point of the task at the same time. The exploration strategy of each tree is the same as that of the basic RRT until the expansion path of the two trees intersects to achieve path planning. However, there is a gap between the planned trajectory and the optimal trajectory, and the quality is relatively poor. Closed-loop RRT (CL-RRT) [33] can realize path planning in a complex environment with multiple obstacles, and the planned path is closer to the optimal path. However, due to the increase in exploration failure points, the planning time is increased, and it is not convenient to address the situation of a complex terrain.

Table 2. The main used path planning methodologies with each advantage and difficulties.

| Path Planning Algorithm     | Advantage                                      | Difficulties                                      |
|-----------------------------|------------------------------------------------|--------------------------------------------------|
| A Star [23]                 | Informed search algorithm high; efficiency in heuristic planning; | Optimal search path cannot be guaranteed in multiple minimum values |
| D Star [24]                 | Incremental search algorithm; rapid planning in re-planning Track planning using random road map method; easy to find the optimal trajectory | Consumes a lot of search and calculation time Cannot be applied to real-time planning Random search is not sensitive to complex environment Cannot focus the search when associated prolate hyperspheroid is large Expensive in high dimension; difficult to find a path in a narrow environment |
| PRM [26]                    | Track planning using random road map method; easy to find the optimal trajectory | Cannot be applied to complex environment |
| RRT [27]                    | Effectively solve high-dimensional space and complex constraints | |
| Informed RRT Star [31]      | Less dependence on the dimension and domain | Cannot focus the search when associated prolate hyperspheroid is large |
| RRT-Connect [32]            | High Search speed and search efficiency | Not convenient to address the situation of a complex terrain |
| Closed-loop RRT [33]        | Well performance in dynamic unstable environment | |

UAVs often move within a dynamic and uncertain environment. This uncertain environment is defined from two aspects: One is that the surrounding environment information detected by the sensor is limited and contains much noise and that the measurement result of the sensor itself is inaccurate. The other is that although we know the global environment information, as time passes, the state of obstacles in the environment will change. Thus, it is necessary to consider the real-time and dynamic nature of obstacles in the environment. The unstructured and complex environment creates great difficulties for robot path planning; thus, local path planning algorithms are generally used. In this paper, we use existing technologies for autonomous exploration and path planning, our contribution performs a new way to combine these technologies to promote the 3D navigation of UAVs in unknown environments.

3. System Overview

The autonomous flight of UAVs in unknown environments relies on a variety of technologies, such as stable and reliable mapping, accurate unknown environment exploration and motion path planning. These issues are related to each other and together constitute a complete autonomous flight system. In this paper, starting from the task of autonomously exploring unknown environments by UAV, combined with frontier-based and sampling-based exploration methods, a voxel implicit frontier clustering model is proposed (See Figure 2). In the study of the path planning, we based on the RRT algorithm, introduced a dynamic step size and an adaptive weight model, and proposed an
advanced RRT algorithm based on the dynamic step size and adaptive weight. To meet the dynamic constraints of the planning path, dangerous flight sections of the planned path are smoothed by Hermite difference polynomials to optimize the planned route. The whole system we proposed called proposed system as Frontier and Sampling-based Exploration and Advanced RRT Planner system (FSEPlanner). This method improves the efficiency of UAV exploration in space, and reduces UAV flight time and optimizes trajectory quality.

![Figure 2](image_url) System overview of our proposed FSEPlanner autonomous exploration and trajectory planning framework.

4. Exploration

This paper proposes a vision autonomous flight system with an independent exploration and path planning in unknown environment. Exploration starts anywhere in the unknown environment. We propose an exploration method combining the frontier and sampling. Frontier is the significant information of exploration; however, original sampling-based exploration algorithm ignores these frontiers. When the next candidate view at the frontier is sampled, it is unnecessary to cluster the voxels into larger frontiers. At the beginning of the exploration, some observations are integrated into the map, as the UAV moves towards the target, the map is constantly updated. When frontier clustering no longer occurs, the exploration is considered completed. Figure 3 gives a graphical overview of the proposed method.

![Figure 3](image_url) Diagram of proposed autonomous exploration structures: Occupancy map, Frontier detection and Candidate position sampling.

4.1. Map Representation

In this work, we utilize occupancy mapping with Supereight to build an environmental map [34]. Supereight exploits an octree structure to store maps. The octree is a kind of tree structure with 8 children in each node, which is suitable for describing the 3D environment. Each node in the tree represents a cube 3D space. The eight sub-nodes of each node describe a subspace. The subspace described by each sub-node can be further divided into sub-spaces and represented by its sub-nodes until the minimum space size is reached. Each layer of the octree is a three-dimensional raster map with different resolutions. The space represented by each node is a voxel of the 3D grid map. The closer the node is to the root node, the larger the voxel size is.
Initially, the UAV extracts 2D slices of a 3D occupied raster based on its fixed height. Each grid cell in the map is classified as “occupied”, “free” or “unknown” according to its occupancy probability. We need a way to convert node coordinates into array indexes. All the nodes in the layer and the children of the nodes should be in continuous memory for easy access and cache-friendly processing (see Figure 4). The Morton order guarantees that the eight sub-nodes of a node are continuous, but there may be discontinuities between these blocks. To store octree nodes in arrays, we need a method to convert their 3D coordinates into 1D array indexes.

![Figure 4](image-url). The work principle of Morton codes. (a) Morton code and traversal ordering for a 2D grid. (b) Schematic diagram of Morton code encoding and indexing.

In the first level of the eight-level structures, the position of each axis is 0 or 1. We can encode each node using one bit per axis, or a total of three bits. When enumerating each node, these bits are incremented as ordinary binary numbers. They can be used as integer indexes for arrays. For larger coordinates, for each additional number on each axis, we continue to have a set of 3 bits, and the Z pattern repeats. The Morton order is faultless for the octree, and the eight children of the node are guaranteed in continuous memory. Octree can contain data in non-leaf nodes. Therefore, each level of the octree has its own array and is filled with the nodes of that level.

Through the use of the Morton code for the spatial index, Supereight is utilized to provide effective map updates and queries. Unlike the leaf level octree, which stores individual voxels, Supereight stores voxel blocks at the leaf level, usually with a size of $8 \times 8 \times 8$ voxels. As new pose and image pairs arrive, the map is constantly updated. We extend Supereight to store whether each voxel is a frontier and perform frontier detection. In addition, the occupancy probability is up-propagated through the octree levels to promote an effective collision query. After the new measurements are integrated into the map, each parent node whose children have been updated as well, and then updates its occupancy probability. The occupation probability of parents is set to the maximum of the occupancy probability of their children.

### 4.2. Frontier Detection

Frontier detection is an edge detection and region extraction techniques from computer vision. Frontier refers to the boundary of a known map. As the robot moves towards the frontier, the known map expands and the boundary expands outward, thus generating a new frontier. This cycle continues until no new frontier is generated, which means that all unknown spaces have been covered by sensors, and it also means the completion of exploration. In frontier-based exploration, robot explores by repeatedly computing (and moving towards) frontiers, the segments which separate the known regions from those unknown.
The map frontiers are constantly updated with the movement of UAV. The obtained map frontiers are stored as a sorted list of Morton codes corresponding to voxel blocks, and the voxels inside the camera frustum are updated with the motion of the UAV. Frontier voxels are seen as free voxels located next to completely unobserved voxels. If the occupancy probability of a voxel is less than 0.5, it is considered a frontier, and one or more of its six adjacent voxels have an occupancy rate of exactly 0.5. As new frontier voxels appear and previously unobserved areas are observed, the Morton code list is updated. When only considering the last updated voxels for the map frontiers update, the whole map area frontier detection can be avoided.

The UAV uses frontier detection to fly to the desired waypoint. Along the way, the occupied map is expanded according to the continuous stereo image. Once the UAV reaches the required frontier or cannot reach the frontier within a certain period of time, the UAV selects another frontier to go towards. Exploration continues until frontier clustering no longer occurs, at which point the exploration is considered complete (see Figure 5).

4.3. Candidate Position Sampling

When frontier detection iteration start, a predetermined number $N_c$ of candidate positions $p_i = [x_i, y_i, z_i]^T \in \mathbb{R}^3$, $i \in \{1 \ldots N_c\}$, are uniformly sampled from the frontier voxel blocks in the frontier list $F$. First, obtain the number of frontier voxels in each frontier voxel block. During the candidate position sampling period, ignored the frontier voxel blocks which the occupancy probability of a voxel is less than 0.5, and defined the filtered frontier list $F'$. Since the frontier voxel blocks $F'$ is sorted according to Morton codes, by utilizing the spatially indexed characteristic of Morton codes, it can be achieving a more uniform candidate position in space by sampling each $[N_{\text{rem}}/N_c]$th frontier voxel block in each $F'$ sampling, where $N_{\text{rem}}$ is the number of frontier voxel blocks in $F$. Coordinated selected frontier voxels and used as a candidate position $p_i$. Robot position during movement is always added to the candidate position to ensure the stability of candidate sampling points at position.

5. Path Planning

Typically, smart drones should have the ability to construct a passable path between two locations in a complex environment. The path planning problem can be defined as follows: Given a description of a mobile platform and its surrounding environment, a dynamic and flexible path that can connect effective starting and ending points is generated between irregular obstacles and meets some optimization indicators, such as the minimum
distance, far-away obstacles, the shortest execution time, the maximum movement speed, and the minimum energy consumption. The path planning algorithm is used to plan a trajectory to the desired frontier if the frontier and sample-based exploration algorithm is selected. Planner uses the current pose estimate and occupancy map.

Rapidly Exploring Random Tree (RRT) is a random path planning algorithm proposed by Steven M. Lavalle [27]. It uses a random function to generate the exploration direction points of tree and selects the point closest to the exploration direction point of the tree as the growth point to carry out the path planning. The RRT algorithm does not require modelling of the physical space but needs to follow the system equations in cybernetics to generate new states in a gradual and incremental manner under the influence of the control variables until the new node state reaches the target point. The planning process of the RRT algorithm has a cyclic structure, which includes three aspects: the process of generating growth points and new nodes and the process of calculating the eigenvalues of the new nodes. The first growth process is shown in Figure 6 below. The beginning and ending are the starting point and target point of the task, $t_{\text{dir}}$ is the random exploration direction point of the tree, $t_{\text{grow}}$ is the growth point of the tree, and $t_{\text{new}}$ is the next new node of the tree to be explored.

![Figure 6. The growth process of the rapidly exploring random tree. ($t_{\text{dir}}$ is the random exploration direction point of the tree, $t_{\text{grow}}$ is the growth point of the tree, and $t_{\text{new}}$ is the next new node of the tree to be explored.)](image)

In principle, RRT takes the starting point of a given task as the root node of the exploration tree and relies on the strategy of building a growing tree in the task environment to complete route planning. It can obtain a path in the mission environment by a random selection strategy and conflict detection method. However, its disadvantage is that it does not know the dispersion of obstacles in the task environment during the exploration process. In complex situations with multiple obstacles, it is easy for the expansion of the exploration tree to fall into a local minimum. Obviously, after the algorithm completes a certain number of iterations, the randomness generated by the direction points causes the efficiency of the algorithm to decrease.

5.1. Advanced RRT

The RRT algorithm still cannot meet the stringent requirements for algorithm performance in path planning in a complex environment. Flight time and path quality are the two main factors for evaluating the pros and cons of the algorithm. This article makes improvements from the following two aspects (see Figure 7).
Figure 7. An overview of the proposed Improved RRT path planning algorithm and trajectory optimization framework.

5.1.1. Improvement in the Selection of the Growth Points

The basic RRT selects the point closest to the tree’s exploration direction point $t_{\text{dir}}$ as the growth point of the tree. This strategy cannot escape quickly when the tree’s exploration falls into the local minimum. In this paper, the suppression factor is adopted. The general idea is that any search failure node $t_1$ in the current tree will have a suppression effect on the tree nodes within a certain range, with the effect of the suppression represented by the suppression factor $I_i$. That is, during the next search of the random exploration tree, the probability of the current search failure node $t_1$ will decrease as a growth point. Similarly, the probability of a certain range of tree nodes acting will decrease as growth points according to the size of the suppression factor $I_i$. The scope of the inhibitory effect is determined by the number of search failures at node $t_1$. If the number of failed searches is relatively small, the scope of the inhibitory effect is relatively small; by contrast, the scope of the inhibitory effect is larger.

Suppose that $m$ tree nodes exist in the current exploration tree $T$, that is, $T = \{t_i | i = 1, 2, \ldots, m\}$. $t_j$ is the search failure node, with $t_j \in T$, and $f_j$ is the number of search failures of $t_j$. If the distance $d_{ij}$ between any node $t_i$ and node $t_j$ results in search failure, $d_{ij} = \|t_j - t_i\|$, the suppression factor $I_i$ of each node $t_i$ in exploration tree $T$ can be calculated by the following formula:

$$I_i = \begin{cases} I / f_j, & d_{ij} < d \\ \frac{d_{ij}}{d^* f_j}, & d \leq d_{ij} < d \times f_j \\ I, & \text{otherwise} \end{cases}$$

(1)

It can be seen from the above formula that if $d_{ij} < d$, the corresponding suppression factor $I_i$ is equal to $I / f_j$. If $d \leq d_{ij} < d \times f_j$, the size of the suppression factor $I_i$ is proportional to $d_{ij}$. If $d_{ij} \geq d \times f_j$, the failed searches node $t_j$ cannot inhibit any current node $t_i$, that is, the size of the suppression factor $I_i$ is 1. The proportional relationship between the size of the suppression factor $I_i$ and the distance $d_{ij}$ between nodes is shown in the Figure 8:
Figure 8. Relationship diagram of suppression factor and nodes distance. \( l_i \) is the size of the suppression factor, \( t_j \) is the search failure node, \( f_j \) is the number of search failures of \( t_j \), \( d_{ij} \) is the distance between any node \( t_i \) and node \( t_j \) results in search failure.

Therefore, in this article, each failed search node \( t_j \) has a certain suppressed effect on other nodes within a certain range. The magnitude of the inhibitory effect is represented by the suppression factor \( l_i \). According to the suppression factor, the weight \( w_i \) of each node is calculated, and then the node with the largest weight is selected as the growth point of the tree to avoid useless exploration, improve the search efficiency of the algorithm, and shorten the planning time.

5.1.2. Dynamic Step Adjustment Strategy

When random exploration tree is extended to the surroundings of the threat source in basic RRT algorithm, the exploration process can easily fall into the local minimum. To avoid the shortcomings of the original RRT, it is necessary to dynamically adjust the exploration step length of the tree to speed up its escape from the local minimum.

As shown in the Figure 9, \( p_k \) represents the \( k \)-th threat source in the mission environment space, and \( \theta \) is the maximum turning angle constraint of the UAV. For example, if it is the growth point of the current tree, the next new node \( t_{i+1} \) to be explored will inevitably fall in the area enclosed by \( t_{\text{grow}} \), \( A \), and \( B \). \( D \) is the current exploration step of the tree, \( \rightarrow l \) is the expansion direction of the current tree, and \( \alpha \) is the angle difference between \( t_{\text{grow}} \) and \( t_{i+1} \) and \( \rightarrow l \). When the line between \( t_{\text{grow}} \) and \( t_{i+1} \) intersects the threat source, the exploration step length of the random exploration tree is adjusted to:

\[
\text{d}' = \frac{\alpha}{\theta} \times d
\]

Otherwise, \( d' = d \).

Therefore, if the expansion of the random exploration tree falls into a local minimum, then if the line between \( t_{\text{grow}} \) and \( t_{i+1} \) intersects the threat source, that is, \( t_{\text{grow}} \) is a failed exploration node, the exploration step of the random exploration tree is adjusted according to the above formula. This process will speed up the escape of the random exploration tree from the local minimum, reduce the number of search failures. When the random exploration tree escapes the local minimum, the exploration step of the exploration tree is restored to the initial exploration step.
Figure 9. The diagram of dynamic step size adjustment chart. \( P_k \) represents the \( k \)-th threat source, \( t_{i+1} \) is the next new node, \( \theta \) is the maximum turning angle constraint, \( \alpha \) is the angle difference between \( t_{\text{grow}}, t_{i+1} \) and \( \rightarrow \).

5.2. Improved RRT Algorithm Structure

Suppose that there are \( m \) tree nodes in the current exploration tree \( T \), that is, \( T = \{ t_i | i = 1, 2, \ldots , m \} \). \( t_j \) is the search failure node, with \( t_j \in T \), and \( f_j \) is the number of search failures of \( t_j \). The distance between any tree node \( t_i \) and the search failed node \( t_j \): \( d_{ij} = \| t_j - t_i \| \), \( d_i \) is the distance between node \( t_i \) and the exploration direction point \( t_{\text{dir}} \): \( d_i = \| t_i - t_{\text{dir}} \| \), and \( I_i \) and \( \omega_i \) are the inhibitory factor and weight of any node \( t_i \) in the tree, respectively. Then, the expansion strategy is as follows:

1. Initialize the exploration tree \( T \), the exploration step \( d_s \), the maximum turning angle \( \theta \), and the turning angle \( \alpha \);
2. Find the random exploration direction point \( t_{\text{dir}} \) (\( t_{\text{goal}} \) and \( t_{\text{random}} \) are the task target point and a random point in the space, respectively, and \( P \) is a random number between 0 and 1).

\[
t_{\text{dir}} = p \cdot t_{\text{goal}} + (1 - p) \cdot t_{\text{rand}}, \ (0 < p < 1) \tag{3}
\]

3. Calculate the exploration step length \( d \):

\[
d = \begin{cases} 
\frac{\theta}{\alpha} \cdot d_s, & f_j > 1 \\
 d_s, & \text{otherwise} 
\end{cases} \tag{4}
\]

4. Select the growth point of the tree \( t_{\text{grow}} \).

\[
I_i = \begin{cases} 
\frac{d_{ij}}{d_i}, & d \leq d_{ij} < d \cdot f_j \\
 I, & \text{otherwise}
\end{cases} \tag{5}
\]

\[
\omega_i = I_i / d_i \tag{6}
\]

\[
t_{\text{grow}} = \text{argmax}(\omega_i), \ t_i \in T \tag{7}
\]

5. Find the new node of the tree \( t_{\text{new}} \)

\[
t_{\text{new}} = \begin{cases} 
t_{\text{dir}}, & \| t_{\text{dir}} - t_{\text{grow}} \| \leq d \\
t_{\text{grow}} + (t_{\text{dir}} - t_{\text{grow}}) / (\| t_{\text{dir}} - t_{\text{grow}} \|), & \text{otherwise}
\end{cases} \tag{8}
\]

6. Determine whether \( t_{\text{new}} \) is a node that has not been explored. If yes, calculate the turning angle \( \alpha \), \( t_j = t_{\text{grow}} \), \( f_j + 1 \), and skip to (2); otherwise, \( t_{\text{new}} \) is added to the exploration tree, \( t_j = \text{NULL} \), and \( f_j = 1 \).
7. Determine whether the target point is reached. If \( \| t_{\text{goal}} - t_{\text{new}} \| < s \), the goal is reached. Otherwise, return to Equation (2). Here, \( s \) is the shortest flight distance of the UAV.

8. Backtrack from the target point to the root node of the exploration tree and return to the planned path.

Algorithm 1 shows the basic framework of improved RRT algorithm. The bold part in the above structure is the improvement of the algorithm expansion, and \( \text{init}(d, \theta) \) is used to initialize the exploration step and the maximum turning angle of the UAV. \( \text{Compute\_Factor}(T, f_j) \) is used to calculate the suppression factor \( l_i \) of the search failure node \( t_j \) to any node \( t_i \) in the tree, \( \text{Compute\_Weight}(T, l_i) \) is used to calculate the weight \( w_i \) of any node \( t_i \) in the tree, and \( \text{Selectmax\_Weight}(T) \) selects the node with the largest weight in the current tree as the growth point \( t_{\text{grow}} \) of the tree. If \( t_{\text{grow}} \) is the search failure node, then \( t_j = t_{\text{grow}} \), and the number of search failures of \( t_j = t_j + 1 \). Use \( \text{Compute\_Angle}(t_{i-1}, t_{\text{grow}}, t_{\text{new}}) \) to calculate the angle between the tree expansion direction and the current heading. \( \text{Compute\_Step}(\alpha, \theta) \) is used to calculate the exploration step length of the tree in the next exploration and to execute the next search. Otherwise, \( t_j = \text{NULL}, f_j = 1 \), and execute the next search.

The basic structure is as follows:

| Algorithm 1 Advanced RRT |
|---------------------------|
| 1. BUILD_ARRT \( (t_{\text{start}}) \) |
| 2. \( T. \text{init}(t_{\text{start}}), \text{Init}(d, \theta) \) |
| 3. \( t_i \leftarrow \text{NULL}, f_j \leftarrow 1 \) |
| 4. \( t_{\text{dir}} \leftarrow \text{Random\_dir}() \) |
| 5. if \( t_i = \text{NULL} \) then |
| 6. \( l_i \leftarrow 1 \) |
| 7. Else |
| 8. \( l_i \leftarrow \text{Compute\_Factor}(T, f_j) \) |
| 9. \( w_i \leftarrow \text{Compute\_Weight}(T, l_i) \) |
| 10. \( t_{\text{grow}} \leftarrow \text{Selectmax\_Weight}(T) \) |
| 11. \( t_{\text{new}} \leftarrow \text{Compute\_New}(t_{\text{dir}} - t_{\text{grow}}) \) |
| 12. if Line \( (t_{\text{grow}}, t_{\text{new}}) \) then |
| 13. \( t_i = t_{\text{grow}} \) |
| 14. \( f_j \leftarrow f_j + 1 \) |
| 15. \( \alpha \leftarrow \text{Compute\_Angle}(t_{i-1}, t_{\text{grow}}, t_{\text{new}}) \) |
| 16. \( d \leftarrow \text{Compute\_Step}(\alpha, \theta) \) |
| 17. Go to step 3 |
| 18. Else |
| 19. \( T.\text{add\_point}(t_{\text{new}}) \) |
| 20. \( T.\text{add\_edge}(t_{\text{grow}}, t_{\text{new}}) \) |
| 21. If \( t_{\text{new}} = t_{\text{goal}} \) then |
| 22. Return \( T \) |

5.3. Trajectory Optimization

The trajectory points of the UAV’s initial trajectory obtained by the algorithm planning are directly connected by a straight line, and the path formed by advanced RRT must have corners at the continuation point, as shown in the left part of Figure 10. If the UAV moves along the original planned path, which will seriously affect the stability of the UAV flight and cause drift. This obviously does not conform to the UAV trajectory in the objective world and violates the UAV constraints on UAVs. After smoothing the trajectory, as shown in the right part of Figure 10, a feasible trajectory meeting the performance constraints of UAV is generated. In this paper, the Hermite difference polynomial [35] is used to optimize the trajectory generated by the RRT algorithm, and finally, a feasible trajectory is obtained.
Figure 10. Schematic diagram of trajectory comparison before (left hand side) and after (right hand side) Hermite difference polynomial trajectory optimization.

Suppose that the current path is T composed of n + 1 trajectory points, with \( t_i \) (\( i = 0, 1, \ldots, n \)) and \( t_0 = S, t_n = D \). Four adjacent tracks are taken at one time point, from which a Hermite curve can be constructed. If the parameter equation \( f(p) \) at the start and end positions \( f(0) \) and \( f(1) \) and the values of the first derivatives \( f'(0) \) and \( f'(1) \) are known, the cubic Hermite can be constructed by the following formula polynomial:

\[
T(p) = (p^3 p^2 p^1 p^0)H \begin{bmatrix}
  f(0) \\
  f'(0) \\
  f(1) \\
  f'(1)
\end{bmatrix} \quad (0 < p < 1)
\]  

\( T(p) \) is the Hermite interpolation polynomial for the function \( f(p) \), and the matrix \( H \) is the Hermite matrix. In this article, to facilitate the realization, we take the following:

\[
f(0) = t(i), f(1) = t(i + 1); 
\]
\[
f'(0) = t(i) - t(i - 1); 
\]
\[
f'(1) = t(i + 2) - t(i + 1); 
\]

\[
H = \begin{bmatrix}
  2 & -2 & 1 & 1 \\
  -3 & 3 & -2 & -1 \\
  0 & 0 & 1 & 0 \\
  1 & 0 & 0 & 0
\end{bmatrix}
\]  

Hermite optimization can eliminate those relatively dangerous and non-flyable trajectories along the original path, avoid flying drift caused by sudden changes in speed at corners, and smoothing and optimization of the planned path (see Figure 11).

Figure 11. An illustration and comparison of the Hermite interpolation polynomial trajectory optimization (before optimization: left hand side, after optimization: right hand side).
This chapter proposes an improved RRT algorithm, aiming at the shortcomings of the original RRT algorithm, which easily falls into a local minimum and long time consuming, this paper improves the selection of growth points and adjustment of the exploration step length and introduces the “adaptive weight” and “dynamic step length”. Advanced RRT path planner is divided into two parts: random exploration tree generation and trajectory smoothing. Through the improvement of growth point selection and step length adjustment, the random exploration tree is generated by combining the planning target and the dynamics constraints of UAV. Smoothing of the original trajectory is carried out to eliminate the dangerous flight sections. Finally, after multiple iterations, the optimal path is output to the UAV navigation system.

6. Experimental Results

6.1. Simulation Setup

A set of simulation studies in apartment environment and maze environment were conducted to evaluate the performance of proposed autonomous exploration method and FSEPlanner. We construct a three-dimensional feasible surface composed of polygonal nodes based on passable and non-passable regions. We use the ground, walls and indoor obstacles as the elements of the 3D virtual scene to build the indoor scene (see Figure 12). In the autonomous exploration experiment, we compared it with the bug algorithm for wall following algorithm proposed in [14], frontier-based exploration proposed in [13] and the receiving horizon NBV planner proposed in [18]. In the path planning experiment, we choose to compare with a*, RRT and VFH+. We use C++ and ROS to build our method. All simulated experiments were performed in Ubuntu 18.04 using ROS Melodic and the Rotor simulator. All simulations were run on an AMD Core R7-5000 CPU and NVIDIA GeForce RTX3070 GPU compiled with g++ 7.4.0.

Figure 12. (a) Orth views and oblique views of apartment simulation environment, (b) Orth views and oblique views of maze simulation environment.
6.2. Apartment and Maze Environment Exploration Simulation Results
6.2.1. Apartment Environment Simulation

We run 25 times simulations in the $5 \times 5 \times 2$ m apartment environment. The proposed autonomous exploration selects a voxel resolution of 0.05 m because this environment contains some narrow corridors, the MAV cannot pass through when such a large resolution is used.

Our algorithm is compared with the bug algorithm for wall following algorithm, frontier-based exploration algorithm and Receiving Horizon NBV planner which we previously mentioned (see Figure 13). As can be seen from Figure 14, our method detects the whole apartment environment much faster. Our method explores 90% and 95% of the environment in 6.7 s and 8.3 s, but Bug algorithms for Wall-following can’t explore 90% of apartment environment (see in Table 3), because Bug algorithms require the planner knows the location of the goal and assume the robot has perfect positioning. When the surrounding environment is sparse, the effect of Frontier based exploration will not decent, so this algorithm cannot explore 95% of the environment. It takes 7.5 s and 11.7 s for receiving horizon NBV to explore 90% and 95% of the environment, respectively. However, random sampling of Receiving Horizon NBV does not always detect unexplored areas quickly. Overall, this method can obtain the best exploration results.

Figure 13. Exploration in apartment environment visualization by (a) Bug algorithm for wall-following, (b) Frontier-Based Exploration, (c) Receding horizon NBV and (d) proposed frontier and sample-based exploration algorithm when voxel resolution of 0.05 m.
6.2.2. Maze Environment Simulation

We run 25 times simulations in the complex maze environment with the size of 10 m × 10 m × 1.6 m. Figure 15 shows the average detection volume of our algorithm and others four autonomous exploration strategy in 25 times using a voxel resolution of 0.1 m, and quantitatively evaluation was shown in Table 4.

In the maze environment, because the environmental area is larger than the apartment environment, a voxel resolution of 0.1 m is selected. In this case, our method performs much faster than the Receiving Horizon NBV algorithm, because the random sampling of the Receiving Horizon NBV algorithm will cause the exploration algorithm to stay in a part of the map for a long time before moving. Our planners did not show this because it used the frontier to reliably guide the MAV to unexplored space. Our method explores 90% of the environment in 23 s, while the receiving horizon NBV algorithm takes 27.4 s. Figure 16 shows the average detection results in 25 runs using four algorithms with voxel resolution of 0.1 m. It should be noted that the calculation time in the large maze is longer, and not all scenes in the maze can be detected by the algorithm due to the occlusion of the wall. Thus, the coverage of this algorithm for this scene is less than 95%. Because the scenes to be explored are larger and naturally involve more frontier clusters, the exploration rate of this algorithm is inferior to that of the Receiving Horizon NBV algorithm. And Frontier-Based Exploration and Bug algorithm for wall-following are inferior to this algorithm in terms of exploration time and exploration coverage.
If we use RRT for direct exploration, in physical space, the robot no longer moves as the random tree grows. Instead, the random tree is used only to detect boundary points independent of the robot’s movement. The detected points will be filtered and sorted for assignment to the robot, and once the point is selected, the robot will move towards it.

Figure 15. Exploration in maze environment visualization by (a) Bug algorithm for wall-following, (b) Frontier-Based Exploration, (c) Receding horizon NBV and (d) proposed frontier and sample-based exploration algorithm when voxel resolution of 0.1 m.

Table 4. The flight time and coverage comparisons of Frontier-Based Exploration, Receding horizon NBV, Bug algorithm for wall-following exploration algorithm with proposed frontier and sample-based exploration in maze environment.

| Algorithm          | Frontier-Based Exploration [13] | Receding Horizon NBV [18] | Bug Algorithm for Wall-Following [14] | Proposed Method |
|--------------------|---------------------------------|---------------------------|--------------------------------------|-----------------|
| Flight time (s)    | 43.59                           | 52.31                     | 47.37                                | 37.69           |
| Coverage (%)       | 87.17                           | 94.75                     | 74.46                                | 93.27           |
Figure 16. Maze environment explored volume over time by four algorithms.

6.3. Apartment and Maze Environment Path Planning Simulation Results

After the exploration of the environment is completed by the boundary and sampling-based exploration algorithm, the advanced RRT algorithm is used to plan the path to reach the desired boundary. In order to evaluate the efficiency of the proposed path generation method, the proposed FSEPlanner method is compared with A* [23], RRT [27] and VFH+ [14] in the apartment as well as in the maze environment.

For the apartment environment, Figure 17 illustrates the results of 25 simulations of trajectory generation using the A* algorithm, RRT and VFH+ algorithms. In this paper, a quadrotor UAV model is constructed and the locations of the initial and target points are given.

Figure 17. Trajectories of five path planning simulation in apartment environment visualization by (a) RRT, (b) A*, (c) VFH+ (d) proposed FSEPlanner without Smoothing and (e) FSEPlanner algorithm.
Figure 17a shows the trajectory generated by the RRT algorithm. RRT is a random search algorithm, which is insensitive to the type of environment. When the space contains a large number of obstacles or narrow channel constraints, the convergence speed of the algorithm is slow and the efficiency decreases significantly. The average flight distance of this algorithm is around 8.15 m (see Figure 18). At the same time, it is not easy to find the optimal path because of the small area of narrow passages. Figure 17b shows the trajectory generated by the quadrotor UAV with the use of the A* algorithm. The A* algorithm uses a heuristic evaluation function to achieve guided search, and the algorithm is efficient and can obtain the shortest path between two points. However, the time and space complexity of the search is high when the scale of the environment is large. The VFH+ algorithm represented in Figure 17c have a decent successful rate in generating paths, but according to the principle and steps of VFH+, it can be seen that it can only output the direction of movement of the next step, and the movement smoothness is poor. Therefore, VFH+ takes about 6.74 s to reach the end point, which takes 0.37 s more compared to FSEPlanner.

![Figure 18](image-url)

Figure 18. Results of the comparison between the RRT, A*, VFH+, proposed FSEPlanner without Smoothing and FSEPlanner algorithm in (a) Distribution of average path length (m), (b) Distribution of average flight time (s) and (c) Average success rate (%) in apartment environment.

Figure 17d,e shows the trajectories generated before and after the optimization process. The trajectories generated before the smoothing process are angular because the map is built using an occupancy grid. The path before optimization affects the smoothness of the flight and increases the endurance and battery consumption.

For the maze environment, Figure 19 illustrates the trajectories generated using the A* algorithm, RRT, and VFH+ algorithm, respectively. RRT generates a new trajectory at each iteration of the algorithm due to the random nature of the algorithm, and the length and time of the generated trajectories perform worse compared to FSEPlanner, and one of the weaknesses of RRT is that it is difficult to find paths in environments with narrow passages. A* algorithm has a shorter search time and higher spatial complexity in the maze environment due to the high environmental complexity, so the processing time of this algorithm is 0.3441 s. VFH+ local planning algorithm does not perform well in global optimization, and it can be seen in the figure that the smoothness of its trajectory in the picture is relatively low. Compared to the other algorithms, FSEPlanner performs best in terms of path length and flight time.
Figure 19. Trajectories of five path planning simulation in maze environment visualization by (a) RRT, (b) A*, (c) VFH+ (d) proposed FSEPlanner without Smoothing and (e) FSEPlanner algorithm.

Next, Figure 20 illustrates the comparison of success rate between the proposed method and other algorithms. The success rate of RRT and A* algorithms decrease as the density of obstacles in the maze environment increases. VFH+ has an okay success rate in replanning by using guided paths, but replanning takes more time, so this algorithm takes longer time. Finally, Figure shows the paths generated before and after the optimization process. The optimized path reduces the flight path of the quadrotor UAV as well as the flight time.

Figure 20. Results of the comparisons between the RRT, A*, VFH+, proposed FSEPlanner without Smoothing and FSEPlanner algorithm in (a) Distribution of average path length (m), (b) Distribution of average flight time (s) and (c) Average success rate (%) in maze environment.
7. Conclusions and Future Work

In this paper, we propose a quadrotor UAV path planning system (FSEPlanner) in an unknown environment combining autonomous exploration and path planning. The method mainly improves the UAV’s ability of autonomous exploration and path planning in unknown environments in terms of flight time consumption and path quality. The method is divided into two main modules: autonomous exploration and path planning.

In autonomous exploration, we combine frontier-based exploration and sample-based exploration algorithms, sampled the frontier voxel blocks according to occupancy probability threshold and evaluated using a utility function combining map entropy and travel time. In path planning section, an improved RRT algorithm is proposed based on the analysis of the fast exploration tree algorithm. To address the shortcomings of the traditional RRT algorithm, which is prone to fall into local minimal areas and take a long time, this paper improves the selection of growth points and the adjustment of exploration steps by introducing “adaptive weights” and “dynamic step size”. Finally, trajectory optimization is used to smooth out the twisted sections of the original path.

Compared to other existing methods of autonomous exploration, due to the targeted sampling, the proposed exploration method needs fewer candidate next-views than sampling-based methods, and by reducing frontier voxel clustering computational cost to speed up the exploration time. In addition, we combine frontier-based exploration and sample-based exploration algorithm to solve the problem of random frontiers generating from traditional frontier-based exploration due to environmental occlusion and noise from the sensors. We implement our algorithm in an apartment and maze simulation environment and compare it with other algorithms. In addition to Receding horizon NBV method, our autonomous exploration algorithm performs better than other algorithms in terms of computing time and coverage. Compared with our frontier and sample-based exploration and Receding horizon NBV method, due to the limitation of computational resources, our exploration method can only generate paths within a limited range by random sampling. The algorithm needs to resample every time the robot moves a certain distance. For each sampling, the algorithm optimizes only the short-term goal. When moving toward the frontier at the current moment, the robot tends to become short-sighted and neglects the long-term goal, thus reducing the overall spatial exploration capability of the algorithm, and the path coverage in the overall region is not high enough compared with Receding horizon NBV. Although the exploration coverage in the maze environment is not as good as that of Receding horizon NBV, the flight time is 27.9% faster than that of Receding horizon NBV. Overall, the autonomous exploration algorithm proposed in this paper enables fast and safe volume exploration in unknown space, ensuring that exploration tasks can be accomplished consistently in complex space environments.

Compared with other existing path planning methods, if only RRT algorithm is used in our work, then path planning in complex scenarios becomes difficult due to the rapidly exploring tree randomly generated. This paper improves the RRT algorithm on the selection of growth points and adjustment of dynamic step. In the simulation experiments of trajectory planning, FSEPlanner outperforms RRT, A*, and VFH+ in both path length, flight time, and success rate of trajectory generation. We also compare the paths generated by FSEPlanner and FSEPlanner before trajectory optimization in different environments, and the results validate the necessity of optimization and the impact of its optimization on path quality.

In this work, proposed FSEPlanner can be directly used for the autonomous navigation of UAV, making it possible for fully autonomous and efficient monitoring. It can be applied to autonomous rescue and monitoring missions. For future work, we will evaluate the proposed approach in real-world experiments with a real mobile robot. Also, fusion of visual sensor and laser sensor data to perform autonomous MAV operation in a complex indoor and outdoor environment.
Author Contributions: Conceptualization, Leyang Zhao and Li Yan; Data curation, Leyang Zhao, Xiao Hu and Jinbiao Yuan; Formal analysis, Leyang Zhao, Xiao Hu and Zhenbao Liu; Funding acquisition, Li Yan; Investigation, Leyang Zhao, Xiao Hu and Zhenbao Liu; Methodology, Leyang Zhao, Xiao Hu, Jinbiao Yuan and Zhenbao Liu; Project administration, Li Yan; Resources, Li Yan and Zhenbao Liu; Software, Leyang Zhao and Jinbiao Yuan; Supervision, Li Yan and Xiao Hu; Validation, Leyang Zhao; Visualization, Leyang Zhao and Jinbiao Yuan; Writing—review & editing, Leyang Zhao. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by National Key Research and Development Program of China OF FUNDER, grant number 2020YFD1100203.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

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

References

1. Shen, S.; Mulgaonkar, Y.; Michael, N.; Kumar, V. Multi-sensor fusion for robust autonomous flight in indoor and outdoor environments with a rotorcraft MAV. In Proceedings of the 2014 IEEE International Conference on Robotics and Automation, Hong Kong, China, 31 May–7 June 2014; pp. 4974–4981.

2. Loianno, G.; Brunner, C.; McGrath, G.; Kumar, V. Estimation, control, and planning for aggressive flight with a small quadrotor with a single camera and IMU. IEEE Robot. Autom. Lett. 2016, 2, 404–411. [CrossRef]

3. Liu, S.; Wattersen, M.; Mohta, K.; Sun, K.; Bhattacharya, S.; Taylor, C.J.; Kumar, V. Planning dynamically feasible trajectories for quadrotors using safe flight corridors in 3-d complex environments. IEEE Robot. Autom. Lett. 2017, 2, 1688–1695. [CrossRef]

4. Papachristos, C.; Kamel, M.; Popović, M.; Khattak, S.; Bircher, A.; Oleynikova, H.; Siegwart, R. Autonomous exploration and inspection path planning for aerial robots using the robot operating system. In Robot Operating System; Springer: Cham, Switzerland, 2019; pp. 67–111.

5. Tang, S.; Kumar, V. A complete algorithm for generating safe trajectories for multi-robot teams. In Robotics Research; Springer: Cham, Switzerland, 2018; pp. 599–616.

6. Shen, S.; Michael, N.; Kumar, V. Obtaining liftoff indoors: Autonomous navigation in confined indoor environments. IEEE Robot. Autom. Mag. 2013, 20, 40–48. [CrossRef]

7. Stumberg, L.; Usenko, V.; Engel, J.; Stückler, J.; Cremers, D. From monocular SLAM to autonomous drone exploration. In Proceedings of the 2017 European Conference on Mobile Robots, Paris, France, 6–8 September 2017; pp. 1–8.

8. Shen, S.; Michael, N.; Kumar, 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.

9. Fraundorfer, F.; Heng, L.; Honegger, D.; Lee, G.H.; Meier, L.; Tanskanen, P.; Pollefeys, M. Vision-based autonomous mapping and exploration using a quadrotor MAV. In Proceedings of the 2012 IEEE/RSJ International Conference on Intelligent Robots and Systems, Vilamoura-Algarve, Portugal, 7–12 October 2012; pp. 4557–4564.

10. Song, S.; Jo, S. Surface-based exploration for autonomous 3D modelling. For autonomy. In Proceedings of the IEEE International Conference on Robotics and Automation, Brisbane, QLD, Australia, 21–25 May 2018; pp. 4319–4326.

11. Dharmadhikari, M.; Dang, T.; Solanka, L.; Loje, J.; Nguyen, H.; Khedekar, N.; Alexis, K. Motion primitives-based path planning for fast and agile exploration using aerial robots. In Proceedings of the 2020 IEEE International Conference on Robotics and Automation, Paris, France, 31 May–31 August 2020; pp. 179–185. [CrossRef]

12. Bachrach, A.; He, R.; Roy, N. Autonomous flight in unknown indoor environments. Int. J. Micro Air Veh. 2009, 1, 217–228. [CrossRef]

13. Yamauchi, B. A frontier-based approach for autonomous exploration. In Proceedings of the 1997 IEEE International Symposium on Computational Intelligence in Robotics and Automation, Monterey, CA, USA, 10–11 July 1997; pp. 146–151. [CrossRef]

14. Heng, L.; Honegger, D.; Lee, G.H.; Meier, L.; Tanskanen, P.; Fraundorfer, F.; Pollefeys, M. Autonomous Visual Mapping and Exploration with a Micro Aerial Vehicle. J. Field Robot. 2014, 31, 654–675. [CrossRef]

15. Connolly, C. The determination of next best views. In Proceedings of the IEEE Conference on Robotics and Automation, St. Louis, MO, USA, 25–28 March 1985; pp. 432–435. [CrossRef]

16. 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, Singapore, 29 May–3 June 2017; pp. 4568–4575. [CrossRef]

17. González-Banos, H.H.; Latombe, J.C. Navigation strategies for exploring indoor environments. Int. J. Robot. Res. 2002, 21, 829–848. [CrossRef]
18. 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, Madrid, Spain, 1–5 October 2018; pp. 1–9.

19. 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, Stockholm, Sweden, 16–21 May 2016; pp. 1462–1468.

20. Dashkevich, A.; Rosokha, S.; Vorontsova, D. Simulation Tool for the Drone Trajectory Planning Based on Genetic Algorithm Approach. In Proceedings of the 2020 IEEE KhPI Week on Advanced Technology, Kharkiv, Ukraine, 5–10 October 2020; pp. 387–390.

21. Ge, J.; Liu, L.; Dong, X.; Tian, W. Trajectory Planning of Fixed-wing UAV Using Kinodynamic RRT* Algorithm. In Proceedings of the 2020 10th International Conference on Information Science and Technology (ICIST), London, UK, 9–15 September 2020; pp. 44–49.

22. Kim, S.; Sreenath, K.; Bhattacharya, S.; Kumar, V. Optimal trajectory generation under homology class constraints. In Proceedings of the 2012 IEEE 51st IEEE Conference on Decision and Control, Maui, HI, USA, 10–13 December 2012; pp. 3157–3164.

23. Hart, P.E.; Nilsson, N.J.; Raphael, B. A Formal Basis for the Heuristic Determination of Minimum Cost Paths. *IEEE Trans. Syst. Sci. Cybern.* 1968, 4, 100–107. [CrossRef]

24. Ferguson, D.; Stentz, A. Using interpolation to improve path planning: The Field D* algorithm. *J. Field Robot.* 2006, 23, 79–101. [CrossRef]

25. Ferguson, D.; Anthony, S. Field D*: An interpolation-based path planner and replanner. In *Robotics Research*; Springer: Berlin/Heidelberg, Germany, 2007; pp. 239–253.

26. Kavraki, L.E.; Svestka, P.; Latombe, J.C.; Overmars, M.H. Probabilistic roadmaps for path planning in high-dimensional configuration spaces. *IEEE Trans. Robot. Autom.* 1996, 12, 566–580. [CrossRef]

27. LaValle, S.M.; Kuffner, J., Jr. Randomized kinodynamic planning. *Int. J. Robot. Res.* 2001, 20, 378–400. [CrossRef]

28. Geraerts, R.; Mark, H.O. A comparative study of probabilistic roadmap planners. In *Algorithmic Foundations of Robotics V*; Springer: Berlin/Heidelberg, Germany, 2004; pp. 43–57.

29. Karaman, S.; Frazzoli, E. Incremental sampling-based algorithms for optimal motion planning. In *Robotics Science and Systems*; Zaragoza, Spain, June 2010; Volume 104.

30. Adiyatov, O.; Varol, H.A. Rapidly-exploring random tree based memory efficient motion planning. In *Mechatronics and Automation (ICMA)*; Nazarbayev University: Takamatsu, Japan, August 2013; pp. 354–359.

31. Gammell, J.D.; Srinivasa, S.S.; Barfoot, T.D. Informed RRT*: Optimal Sampling-based Path Planning Focused via Direct Sampling of an Admissible Ellipsoidal Heuristic. In Proceedings of the 2014 IEEE/RSJ International Conference on Intelligent Robots and Systems, Chicago, IL, USA, 14–18 September 2014; pp. 2997–3004.

32. Kuffner, J.J.; LaValle, S.M. RRT-connect: An efficient approach to single-query path planning. In Proceedings of the IEEE International Conference on Robotics and Automation, San Francisco, CA, USA, 24–28 April 2000; pp. 995–1001.

33. Kuwata, Y.; Teo, J.; Fiore, G.; Karaman, S.; Frazzoli, E.; How, J.P. Real-time motion planning with applications to autonomous urban driving. *IEEE Trans. Control Syst. Technol.* 2009, 17, 1105–1118. [CrossRef]

34. Vespa, E.; Nikolov, N.; Grimm, M.; Nardi, L.; Kelly, P.H.; Leutenegger, S. Efficient octree-based volumetric SLAM supporting signed-distance and occupancy mapping. *IEEE Robot. Autom. Lett.* 2018, 3, 1144–1151. [CrossRef]

35. Spitzbart, A. A generalization of Hermite’s interpolation formula. *Am. Math. Mon.* 1960, 67, 42–46. [CrossRef]