drones

MDPI

Article

An Improved Probabilistic Roadmap Planning Method for Safe Indoor Flights of Unmanned Aerial Vehicles

Qingeng Jin, Qingwu Hu *, Pengcheng Zhao, Shaohua Wang * and Mingyao Ai

School of Remote Sensing and Information Engineering, Wuhan University, Wuhan 430079, China
* Correspondence: huqw@whu.edu.cn (Q.H.); shwang@whu.edu.cn (S.W.)

Abstract: Unmanned aerial vehicles (UAVs) have been widely used in industry and daily life, where safety is the primary consideration, resulting in their use in open outdoor environments, which are wider than complex indoor environments. However, the demand is growing for deploying UAVs indoors for specific tasks such as inspection, supervision, transportation, and management. To broaden indoor applications while ensuring safety, the quadrotor is notable for its motion flexibility, particularly in the vertical direction. In this study, we developed an improved probabilistic roadmap (PRM) planning method for safe indoor flights based on the assumption of a quadrotor model UAV. First, to represent and model a 3D environment, we generated a reduced-dimensional map using a point cloud projection method. Second, to deploy UAV indoor missions and ensure safety, we improved the PRM planning method and obtained a collision-free flight path for the UAV. Lastly, to optimize the overall mission, we performed postprocessing optimization on the path, avoiding redundant flights. We conducted experiments to validate the effectiveness and efficiency of the proposed method on both desktop and onboard PC, in terms of path-finding success rate, planning time, and path length. The results showed that our method ensures safe indoor UAV flights while significantly improving computational efficiency.

Keywords: indoor environment; point cloud; unmanned aerial vehicle; path planning and optimization; probabilistic roadmap

Citation: Jin, Q.; Hu, Q.; Zhao, P.; Wang, S.; Ai, M. An Improved Probabilistic Roadmap Planning Method for Safe Indoor Flights of Unmanned Aerial Vehicles. *Drones* **2023**, *7*, 92. https://doi.org/10.3390/drones7020092

Academic Editors: Nadjim Horri, Samir Khan, Vaios Lappas and Shiva Raj Pokhrel

Received: 9 November 2022
Revised: 7 January 2023
Accepted: 26 January 2023
Published: 28 January 2023

1. Introduction

Small unmanned aerial vehicles (UAVs) have considerably evolved and are increasingly applied in many fields, such as agriculture [1,2], monitoring [3,4], transportation [5,6], delivery [7,8], and rescue [9,10], necessitating additional research on the use of UAVs for mobile robotics, photogrammetry, and monitoring, to name but a few. The core advantage of using UAVs is that they can operate and execute missions in hazardous and dangerous situations. However, certain safety challenges must be considered when integrating UAVs, including attention cost, psychological impact, and physical risks [11]. As such, safety is the most-debated topic of designing and using UAVs [12]. The reason why UAVs are not yet safe, especially indoors, is that they still have flaws such as poor environmental perception and low strain capacity, which means that during autonomous UAV flight in complex environments, safety hazards cannot be completely avoided, posing potential threats to life and property.

Different UAV models, each with their own unique traits, are appropriate for different application scenarios. Among the different models of the UAV, the quadrotor has been extensively developed, researched, and applied over time. The notable quadrotor advantages are its flexibility, adaptivity, and ease of construction [13]. The quadrotor is an aircraft with four rotors with associated propellers, which is capable of hovering, jerking, vertical takeoff and landing, and horizontal flight. Numerous quadrotors are being fabricated for academic research or commercial use, e.g., Pixhawk [14], DraganFlyer X4 [15], and DJI M300 [16], and the boundaries of modeling theories [17–20] and control methods [21–24] are being

advanced. Given that we mainly focused on UAV path planning rather than UAV control in this study, we did not consider the aspect of UAV control. However, several autopilot products or software [25–27] cover the underlying control of UAVs, allowing researchers to perform and evaluate the designed planning methods on supported platforms.

Despite most UAVs being used in open and spacious outdoor situations rather than cramped and unpredictable indoor situations, demand has been growing for deploying UAVs indoors for specific missions, including manufacturing, inspection, and service [28]. Unlike executing a flight outdoors, UAVs cannot use the global navigation satellite system (GNSS) as a location mechanism when indoors. As a result, indoor UAV localization systems typically use a visual camera [29], laser [30], wireless communication [31], or motion capture systems [32] as alternative techniques to assist with indoor tasks. Recently, systems based on new technology such as the Internet of Things (IoT) [33], deep-learning-based monocular cameras [34], and optimal mass flow sensors [35] have been proposed. Additionally, some advanced systems have been developed using more conventional sensors providing solutions for indoor UAV applications, including camera-based tracking systems [36], light detection and ranging (LiDAR)-based systems [37], and multisensor fusion systems [38].

To ensure the safety of UAV mission planning and autonomous flight, especially in indoor environments, UAV systems need to combine local environment awareness with global cognition. By creating a map of the indoor environment, the UAV can realize global cognition of the environment in which it is located, also providing the basis for a series of subsequent operations such as UAV path planning, mission configuration, and autonomous flight. Conversely, when relying solely on the UAV forming local perception of the surrounding environment, only emergency operations can be executed, such as obstacle avoidance, thus passively rather than actively ensuring safe UAV flights at the planning level. To allow UAVs to better understand indoor environments, a commonly used option is the application of payload sensors, where LiDAR provides benefits in illumination tolerance and object structure description. With a point cloud of the scenario as the only output, subsequent processes, such as environment modeling and path planning, can be performed. Based on this, numerous indoor UAV tasks can be accomplished, e.g., non-GPS navigation, interior inspection, and autonomous delivery.

The environmental awareness and cognition of a UAV enable UAV control at the planning level. To complete a UAV flight mission, the first step is deciding where to travel and stop, followed by controls to maintain proper movement during these tasks. Additionally, the UAV needs to be resilient to unforeseen circumstances, especially dynamic obstacles. The latter tasks were outside our scope in this study: our focus was on the process of creating a global traversal path for a UAV in an environment, i.e., path or route planning.

Environment modeling and path search are two major components of path planning. Many methods have been proposed by scholars in these fields, the most important of which are geometric, topological, and cell decomposition methods. The geometric method abstracts the elements of the environment using geometric shapes, such as a convex region combined with polynomial forms [39]. Although these methods can intuitively describe the surroundings, their computational volume noticeably increases when the obstacles are dense and the layout is complicated; additionally, dealing with irregularly shaped obstacles is challenging. With the topological methods, the environment is abstracted as a graph structure, which considerably simplifies the environment and increases path search efficiency, but creating a topological graph is complicated and time-consuming. The core idea behind the cell decomposition method is to divide a large environment into small cells using a simple environment discretization method, which is easy to compute and applies well to situations with many obstacles and complex morphology, reducing the complexity of the environment model and increasing planning efficiency. The grid map [40] is a typical cell decomposition method used in environment modeling.

The purpose of path search is to use specific algorithms to find a path in the environment so that the predetermined performance function returns an optimal value. According

to the basic principle, path search algorithms can be divided into three types: traditional, sample-based, and simulation bionic methods.

Traditional methods were first proposed, but have since been enhanced. Dijkstra's algorithm [41], Floyd–Warshall algorithm [42], A* [43], D* [44], and artificial potential field [45] are a few examples. They are typically simple to calculate and easy to implement, which laid the foundation for resolving the path planning problem. Li et al. [46] proposed a universal path planning method for UAVs that can solve the shortest safe path and a safe least-cost path. González et al. [47] proposed an indoor path-planning algorithm for UAV-based contact inspection that can calculate a path in a few milliseconds. Xu et al. [48] designed a resilient, enhanced algorithm for UAV 3D route planning based on the A* and artificial potential fields algorithms, overcoming the insufficient anti-disturbance ability of the traditional route planning algorithm.

Simulation bionic methods are inspired by natural entities or events, and they achieve their goal by simulating and grouping individual behaviors and interactions, providing a new possibility for solving complex problems. Typical methods include the genetic algorithm [49], ant colony algorithm [50], memetic algorithm [51], etc. The simulation bionic methods are practical for UAV path planning. For example, Liu et al. [52] designed a modified sparrow search algorithm and increased the efficiency in solving the UAV route planning problem. Wang et al. [53] introduced a modified mayfly algorithm, which performed well in the UAV route planning problem. Li et al. [54] combined cellular automata with the spanning tree algorithm to construct a route network in low-altitude airspace, providing a solution for the distribution of logistics UAV. However, although the concept of the simulation bionic methods can be intuitively understood, it has disadvantages in terms of the complexity of description and difficulty in modeling, which do not meet the time requirement and resource limitations of the onboard system.

Sample-based methods involve sampling subgoals in the configuration space to extend the search until the final goal is reached. Rapidly exploring random tree (RRT) [55] and probabilistic roadmap (PRM) [56] are two typical sample-based methods. RRT employs spatial sampling to determine the expansion direction of the search tree. This method can perform fast searches and is adaptable in high-dimensional space; however, it is not well-suited for environments with narrow spaces, and it does not guarantee an optimal path result. PRM, on the contrary, samples available spatial positions serving as nodes in a route network, from which a path search algorithm is used to solve an optimal path from beginning to end. This method converts the path search problem from an entire continuous space to a discrete graph with nodes and edges, making it applicable to path planning problems with high-dimensional and complex constraints. However, because this method is only probabilistically complete, the path result may not be optimal among all possible sample networks.

RRT is more suitable for exploring environments without prior information, whereas PRM is more suitable for selecting a better path in an environment with prior information, so is more suitable for applications that consider the safety of UAV flight. In 2004, Pettersson et al. [57] used the PRM algorithm for an autonomous UAV. Methods based on PRM have been continuously proposed and tend to be combined with other methods. Chen et al. [58] proposed an improved probabilistic roadmap with a potential field function for a quadrotor UAV. Mohanta et al. [59] proposed a knowledge-based fuzzy-probabilistic roadmap for mobile robot navigation. Combining PRM with the A* algorithm is feasible [60], and it works well with the ant colony algorithm [61].

With the development of path planning methods, we have been constantly striving to increase operational effectiveness, reduce computing time, and maintain an optimal path result, to better meet the requirements of safe UAV flight. The sample-based method for path planning achieves an appropriate balance between effectiveness and efficiency. First, this method is well-suited for cell decomposition modeling, where the grid map is a common, mature, and easily used form to describe the environment. Second, as long as any interrelationship exists between cells in the map, it is capable of further simplifying the

map by abstracting it into a graph composed of nodes and edges. Lastly, its path search shows high-quality performance in terms of computational efficiency and path result.

In this study, we focused on improving the computational efficiency of path planning, because it determines whether a UAV can complete the planning and execute an autonomous flight in real-time, i.e., in a few seconds. Due to the fact that indoor environments may be compact and complex, as well as sometimes dynamic and unpredictable, a UAV should be able to finish planning as quickly as possible to handle emergency situations. Based on a typical quadrotor model, we designed an indoor environment reduced-dimensional modeling method that employs point cloud projection to create a downscaled raster map of an indoor environment, reducing the indoor space from 3D to 2D while retaining necessary environmental information such as boundaries and obstacles. We used an adjacency relationship of the grids in a raster map to represent the spatial location relationship in 3D indoor space, thus markedly simplifying the environment. We combined several 2D maps into a multilayer map to produce an improved path-solving result in a complex environment where a single 2D map is not enough to effectively describe the actual situation. Furthermore, we developed an improved PRM planning method, which is an exploratory path search method that converts the path search in indoor environments into a graph search based on sampled nodes. Although the obtained paths may not be the shortest in length due to the sampling randomness, the search capability of the algorithm is remarkably improved, and solving for feasible paths in complex indoor environments is easier. The results of experiments showed that the proposed method substantially reduces the planning time compared with that of the basic PRM algorithm, and it performs well even on a resource-limited computing platform, whereas the postprocessing optimization of the generated paths further improves path quality to meet real-world requirements regarding the timely generation of autonomous UAV flight paths, thereby ensuring UAV flight safety.

2. Generation of Reduced-Dimensional Raster Map Based on Point Cloud Projection

A quadrotor UAV has flexible 3D mobility, i.e., loose constraints on vertical and horizonal motions, which considerably facilitates describing, representing, and modeling an environment. Therefore, we modeled an indoor environment by a point cloud projection method; then, we generated reduced-dimensional raster maps to represent various altitude ranges of the environment, based on which we designed and implemented an improved probabilistic roadmap planning method to obtain mission paths for the UAV. Additionally, we optimized the path by post-process to account for the efficiency and UAV flight safety. An overview of the method workflow is shown in Figure 1.

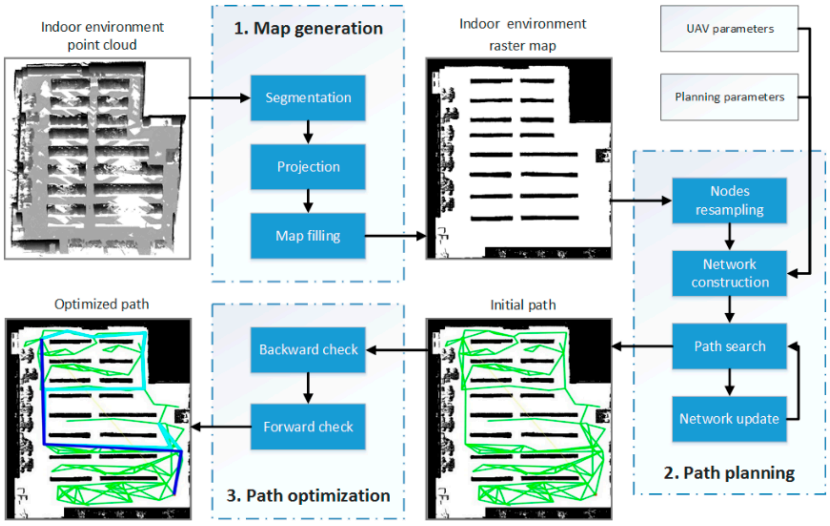


Figure 1. Overview of method workflow.

2.1. Kinematic and Dynamic UAV Model Assumptions

A quadrotor UAV is composed of four rotors attached to the ends of four arms by a symmetric frame. As the direct power source of flight, the rotors can adjust each spinning speed to change the lift force generated by the attached propellers, allowing for flexible horizontal and vertical movements, constant motion, or relative stillness. The control system of a quadrotor is an underactuated system, with six degrees of freedom outputs (three translational motions and three rotational motions) controlled by only four inputs (the spinning speed of four rotors).

In this study, we selected the quadrotor as the assumed type of UAV in the modeling and planning. However, the actual type of the UAV was not our main concern as the current autopilot products and software provide good encapsulation and integration of underlying executions of the UAV, which do not require complex user control. The main reason why we used the quadrotor as the kinematic and dynamic model is that it is capable of flexible mobility in both the vertical and horizontal directions, especially hovering and jerking, which is highly automated by the autopilot.

Some assumptions are required to properly introduce the kinematic model of a quadrotor UAV. We assume that it has a symmetric and rigid structure, with propellers of equal height on the rotors, and the mass center is the same as the space center of the UAV.

We first define two coordinate systems: body inertial frame $O_bX_bY_bZ_b$ and fix inertial frame $O_fX_fY_fZ_f$. The center O_b is defined the same as the mass center of the UAV, with the forward and upward directions of the UAV being the x-direction and z-direction, respectively; the y-direction is determined by the right-hand rule. For simplicity, the fixed inertial frame has the same definition as the first body inertial frame of the UAV, and it does not change once determined.

As such, the UAV state q^T in the environment can be described as

$$q^T = \{x, y, z, \psi, \theta, \phi\}, \tag{1}$$

where (x, y, z) denotes the position of the center of the quadrotor in a fixed frame, and (ψ, θ, ϕ) denotes the orientation of the quadrotor in the body frame represented in Euler angles (yaw, pitch, and roll), which can be further transformed to the fixed frame by

$$R_b^f = \begin{bmatrix} \mathrm{c}(\psi)\mathrm{c}(\theta) & \mathrm{c}(\psi)\mathrm{s}(\theta)\mathrm{s}(\phi) - \mathrm{s}(\psi)\mathrm{c}(\phi) & \mathrm{c}(\psi)\mathrm{s}(\theta)\mathrm{c}(\phi) + \mathrm{s}(\psi)\mathrm{s}(\phi) \\ \mathrm{s}(\psi)\mathrm{c}(\theta) & \mathrm{s}(\psi)\mathrm{s}(\theta)\mathrm{s}(\phi) + \mathrm{c}(\psi)\mathrm{c}(\phi) & \mathrm{s}(\psi)\mathrm{s}(\theta)\mathrm{c}(\phi) - \mathrm{c}(\psi)\mathrm{s}(\phi) \\ -\mathrm{s}(\theta) & \mathrm{c}(\theta)\mathrm{s}(\phi) & \mathrm{c}(\theta)\mathrm{c}(\phi) \end{bmatrix}, \tag{2}$$

where $\mathrm{c}()$ and $\mathrm{s}()$ denote cos and sin operators, respectively.

To further simplify the dynamic model for ease of implementation, we overlook pitch and roll controls for now, because they strongly impact the flight stability of the UAV, and delegating these controls to the autopilot would be preferable. Translational and yaw controls are necessary, as the former is used to change the spatial position of the UAV, while the latter is used to adjust the heading direction. Thus, the following control states remain:

$$\dot{q}^T = \left\{\dot{x}, \dot{y}, \dot{z}, \dot{\psi}\right\}, \tag{3}$$

where $\{\dot{x}, \dot{y}, \dot{z}\}$ denotes the respective speeds with reference to the fixed frame, and $\dot{\psi}$ denotes the change rate of the yaw angle in the body frame.

These speed control parameters are inputs to the UAV kinematic model to keep the UAV on the resulting path solved at the planning level; they are also outputs of the UAV dynamic model, where traction and torque are inputs. However, because the implementation of dynamic modeling can be delegated to the autopilot and we focused more on planning methods than control methods for the UAV, we do not provide further discussion on this topic.

2.2. Indoor Environment Rasterization

In simple indoor environments, describing the inter-relationships of boundary surfaces and obstacle shapes is relatively simple, facilitating the vectorization of environmental elements. However, for complex indoor environments, the traditional vectorization method has limitations. For example, when a room is irregularly shaped, an increase in the number of walls causes the constraints of the boundary to become more complex, and more parameters must be added to the model to completely describe the entire environment, which also substantially affects the efficiency of modeling.

We implemented a downscaling modeling method for indoor environments based on point cloud projection that avoids the vectorization of environment elements and generates a reduced-dimensional raster map based on point cloud coordinate values. The method converts the original three-dimensional space to two-dimensional space and transforms the spatial location relationship between environment elements into the adjacency relationship between elements in the raster map, which considerably reduces the complexity of modeling, improves efficiency, and is more compatible.

The reduced-dimensional raster map consists of small elements called grids, each of which represents a specific size in space. They can be classified based on their values to distinguish boundaries, obstacles, and free space in the environment. Due to the fact that both the boundaries and obstacles are impassable on a map, they can be represented and grouped together as obstacle grids. In addition to obstacle grids, free grids exist in the map, which composes the entire set of map grids.

If we add an attribute and set a value of the free grid $value_{free} = class0$ and the obstacle grid $value_{obstacle} = class1$, we obtain a simple environment reduced-dimensional raster map, as shown in Figure 2, which is essentially a binary image with a size of $height * width$.

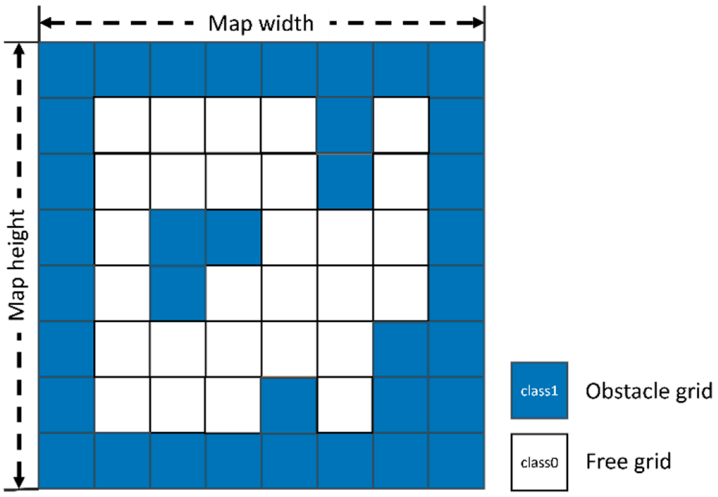


Figure 2. Environment reduced-dimensional raster map.

2.3. Indoor Environment Point Cloud Projection

To preserve the relationship between obstacles and free space as much as possible throughout the conversion of a 3D point cloud to a 2D map, we adopt the indoor environment point cloud projection method, which projects the target 3D point cloud onto a parametric model. In this study, we projected the 3D point cloud of the indoor environment onto a horizontal plane along the vertical direction to provide a vertical view that serves as a reduced-dimensional raster map model of this indoor environment.

In addition, because the UAV has a limited field of vision and concentrates on a specific region rather than a global one, only the points within a range close to the UAV are projected. Thus, we further determine the range for projection according to the detection distance of the UAV:

$$\begin{aligned} x_{min} &= x_{UAV} - d_{UAV},\ x_{max} = x_{UAV} + d_{UAV}, \\ y_{min} &= y_{UAV} - d_{UAV},\ y_{max} = y_{UAV} + d_{UAV}, \\ z_{min} &= z_{UAV} - h/2,\ z_{max} = z_{UAV} + h/2, \end{aligned} \tag{4}$$

where x_{min}, x_{max}, y_{min}, y_{max}, z_{min}, and z_{max} denote the projection range; $(x_{UAV}, y_{UAV}, z_{UAV})$ denotes the UAV position (or another specified center position of the focused point cloud); d_{UAV} denotes the detection range of the UAV and h denotes the specified range for altitude. All the variables are in the same local coordinates as the point cloud.

Any 3D point $p(x, y, z)$ in the point cloud satisfies

$$\begin{aligned} x_{min} < x < x_{max}, \\ y_{min} < y < y_{max}, \\ z_{min} < z < z_{max}. \end{aligned} \tag{5}$$

The conversion from 3D point p to 2D map grid p' is as follows:

$$p' = \begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} round(x - x_{min}) \\ round(y - y_{min}) \end{bmatrix} * s, \tag{6}$$

where $round()$ denotes the rounding sign to resample the 3D point to a 2D point and s denotes the map resolution scaling factor.

The reason why we chose a vertical projection is that indoor objects, in most cases, are vertically placed on the floor, and the free space also extends in the vertical direction. The vertical view is the most widely used map form in robot mapping and navigation applications, reflecting its usefulness, effectiveness, and representativeness.

However, the projection method may disregard the vertical structure of obstacles, particularly in complex environments. To overcome this issue, we further vary the z_{min} and z_{max} in the projection and construct grid maps representing the free space and obstacles at various altitudes. For example, a multilayer grid map $M\{m^1, m^2, \ldots, m^n\}$ consists of grid maps at various altitudes, and the ith map m^i is formed by the projection of point cloud P^i:

$$\begin{aligned} &\forall p^i(x^i, y^i, z^i) \in P^i, \\ &x_{min} < x^i < x_{max}, \\ &y_{min} < y^i < y_{max}, \\ &z^i_{min} < z^i < z^i_{max}, \end{aligned} \tag{7}$$

where z^i_{min} varies from $(z_{UAV} - nh/2)$ to $(z_{UAV} + nh/2)$ with a step size of h.

As a result, the UAV can search for a path not only in one single map but also by merging maps at various altitudes if necessary. In simple cases, a grid map of the near range of the UAV altitude is sufficient to solve a feasible path; in more complex cases, we first search for a path at the current altitudes, and if it is not feasible, we continue searching for subsequent paths in adjacent maps from where the initial search ended. As such, we achieve an appropriate balance between effectiveness and efficiency.

2.4. Indoor Environment Reduced-Dimensional Raster Map Generation

In the reduced-dimensional raster map of indoor environments, the map grid is divided into two categories: obstacle and free grids. The obstacle grids represent two types of environmental elements: boundaries and obstacles. Before constructing a map, the original point cloud of the indoor environment must be preprocessed to generate a usable map of the indoor environment.

Preprocessing commonly comprises the segmentation of floor and ceiling points, denoising, and other processes. Removal of the floor and ceiling points is necessary; otherwise, they obscure the location of the free space and cover the entire projection surface. We used random sample consensus (RANSAC) [62] to segment and extract the floor and ceiling points. RANSAC not only satisfied our segmentation requirements, but also provided us with the parameters of the extracted planes that could be used to determine the

UAV altitude in the environment. Based on this, we divide the space into varying altitudes and generate a map at each altitude.

The process of generating the reduced-dimensional raster map is shown in Figure 3. More specifically, the procedure entails the following steps:

1. Extract the floor and ceiling points using point cloud segmentation, and remove them from the original point cloud. Segmentation range and height can be specified.
2. Calculate the maximum and minimum values of the remaining point clouds on the X and Y axes for the height and width of the map image, respectively. Scaling up of the image is optional to increase model accuracy.
3. Iteratively read the 3D position values of each point, and convert them into map grid coordinates by projection.
4. Repeat step 3 until all points have been traversed and obtain a binary image of the raster map.
5. Vary the specified height of projection to generate raster maps at various altitudes.

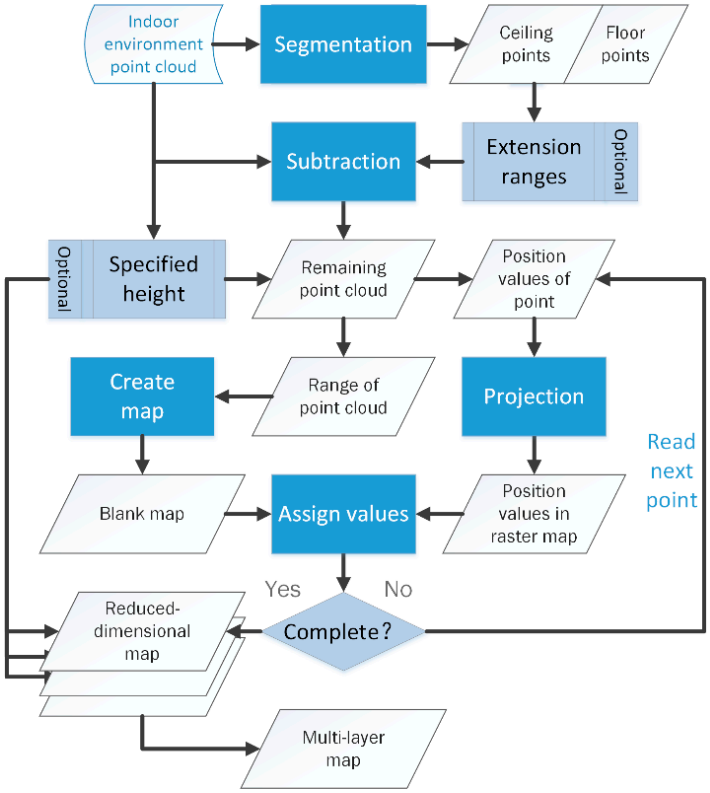


Figure 3. Workflow of the method used to generate reduced-dimensional raster map of indoor environment based on point cloud projection.

3. Improved Probabilistic Roadmap Planning for Safe UAV Flight

3.1. Basic PRM Algorithm

The PRM algorithm, which is essentially a graph-based path search method, is based on the fundamental concept of randomly generating sampling points in free space that serve as graph nodes. After verifying the connectivity of nodes and constructing a connection network, the PRM algorithm conducts a search and then solves a path from the source to the goal.

The PRM algorithm considerably simplifies the environment by discretizing the space into a graph, and is applicable to high-dimensional spaces with complex constraints. However, it is time-consuming and inefficient in network initialization. Additionally, its stability is restricted by the number of sampling nodes and their random locations. The algorithm is therefore probabilistically complete.

The workflow of the basic PRM algorithm mainly includes three parts: spatial sampling, edge generation, and path search. The pseudo-code for the basic PRM is shown in Figure 4. More specifically, the procedure entails the following steps:

(1) Define a node set N; add the source node n_{src} and the goal node n_{goal}.
(2) Generate a node n_{rand} by random sampling in the entire map.
(3) Perform a collision check on n_{rand}. If it passes, add n_{rand} to N; otherwise, return to step 2.
(4) Repeat steps 2 and 3 until M nodes in total have been generated, completing spatial sampling.
(5) Define an edge set E.
(6) Traverse n_m in N and select other nodes n_k to generate edge $e_{m,k}$; perform a collision check on it. If it passes, add it to E.
(7) Repeat step 6 until all M nodes have been traversed, completing edge generation.
(8) Define a graph $G(N, E)$ and deploy a path search algorithm to solve the shortest path P from n_{src} to n_{goal}, completing the path search.

Algorithm 1 Basic Probabilistic Roadmap. It takes two set of node coordinates determining source n_{src} and goal n_{goal} positions and number of sampling nodes M. The algorithm process includes three processes, SAMPLE(), CREATE_EDGES() and FIND_PATH().

SAMPLE(n_{src}, n_{goal}, M):

Input: Node coordinates of source n_{src} and goal n_{goal}, number of sampling node M.

N.add(n_{src}, n_{goal});
N.add(n_{goal});
while N.size() < M **do**
n_{rand} ← RANDOM_STATE();
if CHECK_COLLISION(n_{rand}) == COLLISION_FREE
N.add_node(n_{rand});
return N

CREATE_EDGES(N):

Input: A set of sampling nodes N.

for m = 1 **to** N.size() **do**
for k = 1 **to** N.size() **do**
if n_m != n_k
$e_{m,k}$ ← EDGE(n_m, n_k);
if CHECK_COLLISION ($e_{m,k}$) == COLLISION_FREE
E.add_edge($e_{m,k}$);
return E

FIND_PATH(N, E, n_{src}, n_{goal}):

Input: A set of sampling nodes N and a set of connection edges E, Node coordinates of source n_{src} and goal n_{goal}.

G.init(N, E);
P ← PATH_SOLVE(G, n_{src}, n_{goal});
return P

Figure 4. Pseudocode of basic PRM.

3.2. Improvement Strategies for PRM Algorithm

The basic PRM algorithm has disadvantages in terms of stability and efficiency.

Its insufficient stability is caused by its reliance on the number of sampling nodes and their random locations. When the number of randomly generated nodes in the space is small, or the distribution is unfavorably located, as shown in Figure 5a,c, it may fail to form a network connecting the source and goal, instead generating several disconnected local networks. Nevertheless, the PRM algorithm is probabilistically complete, which means that as long as the random nodes are distributed throughout the space, a feasible path must be found. Therefore, the stability issue can be mitigated by appropriately increasing the number of nodes according to the complexity of the actual indoor environment.

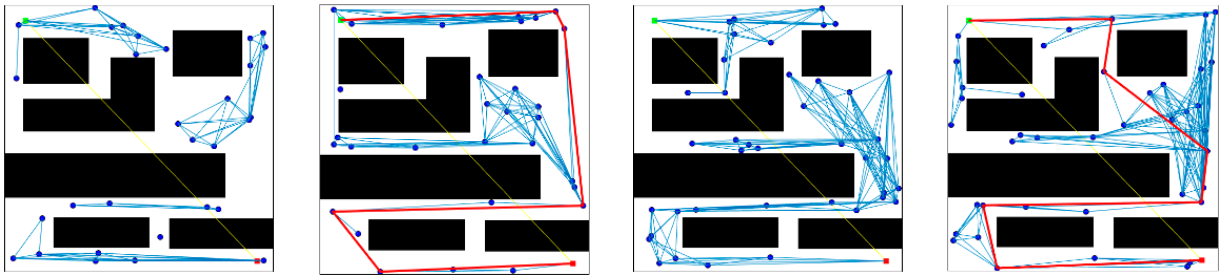


Figure 5. Node distribution under different node numbers (n): (**a**) $n = 30$, undesirable distribution; (**b**) $n = 30$, desirable distribution; (**c**) $n = 40$, undesirable distribution; (**d**) $n = 40$, desirable distribution.

The inefficiency is that some steps in the algorithm, particularly edge generation, are time-consuming. Each edge necessitates a collision check during generation to ensure a collision-free network. Furthermore, as the distance between nodes increases, the likelihood of obstacles between them increases, and generating a collision-free edge becomes more difficult. An effective solution is to reduce the number of collision checks and edge generations between distant nodes to improve the efficiency of the algorithm while having less impact on network connectivity.

To further reduce edge collision checks, we can adopt a strategy of constructing first and checking later, i.e., we do not perform the collision check on every pair of nodes in the process of edge generation after spatial sampling, but perform the collision check after solving a candidate path. Moreover, we eliminate the infeasible edges in the candidate path and find a new path that can reconnect the remaining edges. This strategy restricts the collision check of all edges to only the candidate path and its neighboring nodes and edges, therefore substantially lowering the number of collision checks and improving the efficiency of the algorithm.

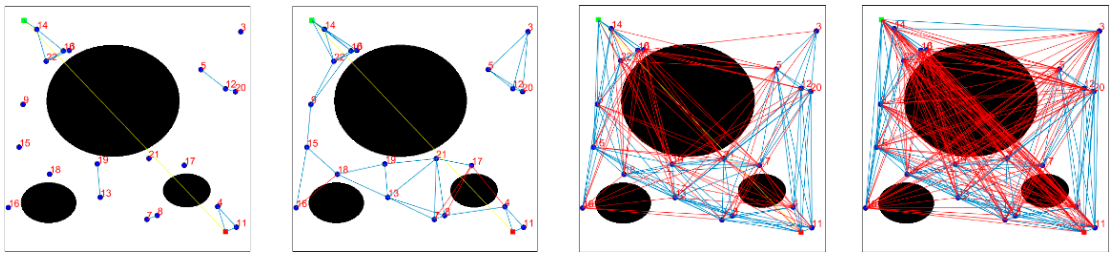
3.3. Network Construction Based on Connection Distance

As the distance between nodes increases, the likelihood of obstacles between them increases, resulting in their invisibility and the impossibility of constructing collision-free edges. On the basis of this insight, we developed a method of network construction based on connection distance.

First, we set a "connection distance" parameter to determine whether to generate a connection edge between two nodes. During the edge generation process, the distance between each node is calculated when traversing each node to the other nodes. If it is above the threshold, the edge is not connected, and the subsequent collision check is skipped; otherwise, the collision check of the edge is performed again, and if it passes, a connected edge is generated between them.

The moderate connection distance is important. If the connection distance is too large, many colliding edges are still unnecessarily checked; however, if the connection distance is too small, network connectivity may be reduced or the network may become disconnected, which will affect the subsequent path search results, as shown in Figure 6.

This method improves the efficiency of the algorithm with little impact on the network connectivity by reducing the connection of nodes whose distance exceeds an acceptable threshold. Comparative networks with different connection distances *cdis* are shown in Figure 6, where the number of collision-free edges e_{free} and the number of colliding edges $e_{collided}$ are counted. As the connection distance increases, the number of colliding edges rapidly increases, while the number of collision-free edges slowly increases. This demonstrates that a proper connection distance can effectively reduce invalid checks for the colliding edges while assuring minimal disruption of network connectivity.



(a) $e_{free} = 13$, $e_{collided} = 0$. **(b)** $e_{free} = 36$, $e_{collided} = 4$. **(c)** $e_{free} = 82$, $e_{collided} = 69$. **(d)** $e_{free} = 92$, $e_{collided} = 139$.

Figure 6. Networks with connection distances of 0.1 (**a**), 0.2 (**b**), 0.5 (**c**), and 1 (**d**), consisting of collision-free edges (e_{free}) indicated by blue lines and colliding edges ($e_{collided}$) by red lines. With increasing connection distance, the number of colliding edges rapidly increases, while the number of collision-free edges slowly increases.

3.4. Path Local Check and Incremental Update

Based on the strategy of constructing first and checking later, we propose a method for path local checking and incremental updating.

After spatial sampling of the nodes, a network is constructed based on the connection distance, but no edge check is conducted at this stage. Hence, colliding edges may exist between the invisible nodes in the network, for which we then conduct a path search from the source to the goal for an initial path. On this path, we execute a minimum number of collision checks by incremental update. If an edge is collided, preventing direct passage between two nodes on the edge, we remove it from the network and search for a new path connecting the two nodes. These steps are repeated until all edges on the path pass the collision check, i.e., the entire path satisfies the no-collision requirement.

The workflow of the improved PRM algorithm mainly includes two parts: network initialization and path update. The pseudocode of the improved PRM is shown in Figure 7. More specifically, the procedure entails the following steps:

(1) Define a node set N; add the source node n_{src} and the goal node n_{goal}.
(2) Generate a node n_{rand} by random sampling in the entire map.
(3) Perform a collision check on n_{rand}. If it passes, add n_{rand} to N; otherwise, return to step 2.
(4) Repeat steps 2 and 3 until M nodes in total have been generated.
(5) Define an edge set E.
(6) Traverse n_m in N and select other nodes n_k to generate edge $e_{m,k}$; add it to E without performing collision checking.
(7) Repeat step 6 until all M nodes have been traversed, completing network initialization.
(8) Define a graph $G(N, E)$, and traverse a current node n_{cur} starting from n_{src}.
(9) Find the nearest neighbor n of n_{cur} and perform a collision check on $e_{n_{cur},n}$. If it passes, add n to result path P and move n_{cur} backward to n; otherwise, remove n from G and update the network.
(10) Repeat step 9 until reaching n_{goal}, completing path update.

The methods of local path check and incremental update essentially comprise a path search strategy that reduces ineffective collision checks on edges, decreasing the time required and increasing algorithm efficiency. The method has strong applicability in environments that are not extremely complex, and a feasible path can be quickly solved with minimal redundancy in simple environments. The procedure of local check and incremental update is shown in Figure 8. For colliding edges (red line) in the initial path, a new path (yellow lines) connecting two segments of the initial path is searched, which serves as a newly available local path for a final collision-free path (green lines).

Algorithm 2 Improved Probabilistic Roadmap. Similarly to the Basic Probabilistic Roadmap, it takes two sets of node coordinates determining source n_{src} and goal n_{goal} positions and number of sampling nodes M. The algorithm process includes **INITIALIZE_NETWORK()** and **UPDATE_PATH()**.

INITIALIZE_NETWORK(n_{src}, n_{goal}, M):

Input: Node coordinates of source n_{src} and goal n_{goal}, number of sampling node M.

N.add(n_{src}, n_{goal});
N.add(n_{goal});
while N.size() < M **do**
n_{rand} ← RANDOM_STATE();
if CHECK_COLLISION(n_{rand}) == COLLISION_FREE
N.add_node(n_{rand});
for m = 1 **to** N.size() **do**
for k = 1 **to** N.size() **do**
if n_m != n_k
$e_{m,k}$ ← EDGE(n_m, n_k);
E.add_edge($e_{m,k}$);
return N, E

UPDATE_PATH(N, E, n_{src}, n_{goal}):

Input: A set of sampling nodes N and a set of connection edges E, Node coordinates of source n_{src} and goal n_{goal}.

G.init(N, E);
n_{cur} ← n_{src};
while n != n_{goal} **do**
n ← NEAREST_NEIGHBOR(G, n_{cur});
$e_{n_{cur},n}$ ← EDGE(n_{cur}, n);
if CHECK_COLLISION ($e_{n_{cur},n}$) != COLLISION_FREE
G.remove_edge($e_{n_{cur},n}$);
else
P.add_node(n);
n_{cur} ← n;
return P

Figure 7. Pseudocode of improved PRM.

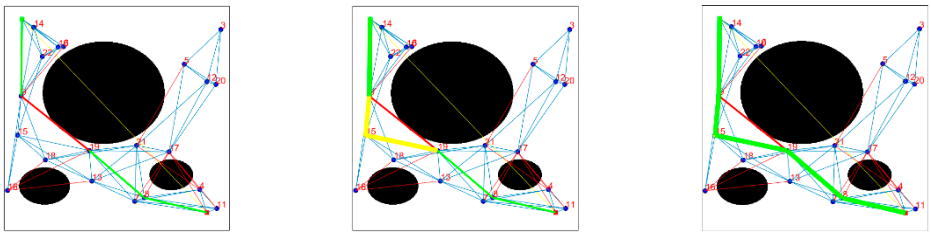
3.5. Path Planning in Multilayer Grid Map

In more complex cases, a single map at a certain altitude might not be appropriate for solving a feasible path if obstacles are blocking the map and dividing it into several disconnected areas. Although areas in a single map may not be connected at the same

altitude, they may be connected via another area at a different altitude. Therefore, we use a multilayer grid map for path planning.

Path search and update strategies in the multilayer map are quite similar to those in a single map; however, the essential distinction lies in how the transfer areas (overlapping areas of adjacent layers) are determined for the UAV to adjust its altitude.

To detect available transfer areas, we first use an image region-growing algorithm to identify and segment the disconnected areas in each single map. Due to the fact that the total number of the areas is uncertain, we randomly sample the growth seeds on the map. If the growing region contains a sufficient number of grids, we record it as a valid area and then continue to sample a new seed and search other areas until all valid areas have been segmented. Furthermore, we examine the connectivity between areas in each layer and those in the adjacent layers. If two areas have grids with the same X and Y coordinates, they are regarded as connected, and the overlapping area formed by the grids is considered to be a transfer area.



(a) Initial path **(b)** Check and update **(c)** Collision-free path

Figure 8. Path local check and incremental update: (**a**) an initial path contains collision-free edges (green lines) and a colliding edge (red line); (**b**) a new path (yellow lines) connecting two segments of the initial path is searched and updated; (**c**) a collision-free path is finally completed by adding the newly available local path.

For the path search process, we attempt to find a path from the source to the goal in a single map at the default altitude. If the path search fails, meaning that impassible obstacles may be located at this altitude, we search in its adjacent layers for any area that overlaps the current search area. If several overlapping areas exist, we use a greedy strategy to select the area with the smallest horizontal distance from the goal as the next search area. In the selected overlapping area, we additionally sample a transfer node for the UAV adjusting its altitude, which also serves as a temporary goal node of the current search area and the source node of the next search area. Thus, we accomplish the cross-layer path search for UAVs in complex indoor environments.

The pseudo-code of path planning in the multilayer map is shown in Figure 9. More specifically, the procedure entails the following steps:

Define $Map_{current}$ as the currently used map (at altitude h_{start} by default) and n_{sub_src} and n_{sub_goal} as the source and goal for the current search, respectively; they are initialized in n_{src} and n_{goal} at first.

(1) Define and initialize node set N as the network nodes in $Map_{current}$.
(2) Start the search for a path from n_{sub_src} to n_{sub_goal} on network N.
(3) If the search fails, find another map Map_{next} that has the smallest distance with n_{sub_goal} as the next map. Search the transfer area of the two maps and sample a transfer node $n_{transfer}$ in it, and set n_{sub_goal} to $n_{transfer}$. Start a new search in this configuration.
(4) If the search succeeds, record the path result in P, switch the map to Map_{next} and n_{sub_src} to $n_{transfer}$, and reset n_{sub_goal} to n_{goal}.
(5) Repeat steps 4 to 5 until P contains n_{src} and n_{goal}.

Because our method is based on the pre-captured point cloud of the environment, the proposed method of map area detection considers a global perspective, i.e., with a priori

knowledge of the environment. Possible enhancements for UAV discovery in an unknown environment were beyond the scope of this study.

Algorithm 3 Path Planning in Multi-Layer Grid Map. In transfer area identification, it takes each layer in the multi-layer grid map $Maps$ and split it into several disconnected areas, based on which it extracts overlapping areas. In path search, it takes two set of node coordinates determining source n_src and goal n_goal positions, number of sampling nodes M and UAV altitude h_{start}. The algorithm process includes **SEGMENT_AREAS()**, **EXTRACT_OVERLAPS()** and **PATH_SEARCH_MULTI_LAYERS()**.

SEGMENT_AREAS($Maps$):

Input: Multi-layer map $Maps$.

for map **in** $Maps$ **do**
  while $seed$!= NULL
    $seed \leftarrow$ RANDOM_FREE_GRID();
    $a \leftarrow$ REGION_GROWING($seed$);
    $Areas\{map, i\}$.add_area(a);
    i++;
return $A \leftarrow Areas$

EXTRACT_OVERLAPS(A):

Input: Segmented areas A in Multi-layer map.

for $map1$ = 1 **to** A.size() **do**
  for $map2 = map1.lower\_layer()$ **to** $map1.upper\_layer()$ **do**
    $freeGrids \leftarrow$ AND($map1$.free_grids(), $map2$.free_grids());
    $transferAreas\{map1, map2\} \leftarrow$ INTERSECT($freeGrids$, $map1$);
return $T \leftarrow transferAreas$

PATH_SEARCH_MULTI_LAYERS($Maps$, T, M, n_{src}, n_{goal}, h_{start}):

Input: The multi-layer map $Maps$, a set of transfer areas T, node coordinates of source n_{src} and goal n_{goal}, number of sampling node M and default UAV altitude h_{start}.

$Map_{current} \leftarrow Maps$.get_map($h_{start}$);
$n_{sub\_src} \leftarrow n_{src}$;
$n_{sub\_goal} \leftarrow n_{goal}$;
while !(P.contain(n_{src}) **&&** P.contain(n_{goal})) **do**
  $N \leftarrow$ INITIALIZE_NETWORK($Map_{current}$, n_{sub_src}, n_{sub_goal}, M);
  if PATH_SOLVE($Map_{current}$, N, n_{sub_src}, n_{sub_goal}) != SUCCESS
    $Map_{next} \leftarrow$ MIN_DISTANCE($T\{Map_{current}\}$, n_{sub_goal});
    $n_{trasnsfer} \leftarrow$ SAMPLE($T\{Map_{current}, Map_{next}\}$);
    N.add_node($n_{trasnsfer}$);
    $n_{sub\_goal} \leftarrow n_{trasnsfer}$;
  else
    $P \leftarrow$ PATH_SOLVE_RESULT($Map_{current}$, N, n_{sub_src}, n_{sub_goal});
    $Map_{current} \leftarrow Map_{next}$;
    $n_{sub\_src} \leftarrow n_{transfer}$;
    $n_{sub\_goal} \leftarrow n_{goal}$;
return P

Figure 9. Pseudocode of path planning in multi-layer grid map.

4. Path Postprocessing Optimization

The randomness in the sampling nodes of the PRM algorithm may result in redundant nodes in the path, which manifests as unnatural distortions in path morphology, as well as multiple visits to nodes during path checking and updating, thus increasing the final path length. In this regard, the initial path obtained from the path search should be optimized by postprocessing to remove redundant nodes and avoid unnecessary visits to the same node, thereby improving path quality and increasing the safety of UAV flight along the path.

In this study, we developed a two-step postprocessing method for path optimization, consisting of a backward and a forward path connection check. The core concept of the method is to search in the initially solved path for a set of "key nodes", forming an optimal path that is collision-free with obstacles and as short a length as possible.

4.1. Backward Path Connection Check

To reduce redundant nodes, the backward connection check method starts from the source node as the first determined node and searches backward, node by node, to the farthest visible node as its direct connection node, which is also the newly determined node. To avoid multiple visits to the same node, each time the node is determined, the initial path is queried to discover if other nodes are visible. If so, it jumps to the last visible node and removes any other nodes in between.

The pseudocode of the backward path connection check is shown in Figure 10. More specifically, the procedure entails the following steps:

(1) Initialize the optimal path P' by adding n_{src}.
(2) Read the initially solved path P, starting from n_{src} as determined node n, with the neighboring node behind as the displacement node n_m.
(3) Check if P contains more than one n, i.e., multiple visits occur to n. If so, jump to the last occurrence of n on P, set it as the newly determined node n, and update the displacement node n_m; otherwise, continue to the next step.
(4) Perform collision check on the edge $e_{n,m}$. If it passes, let n_m move backward; otherwise, add n_{m-1} to P' and update n to n_{m-1} and n_m to $n+1$.
(5) Repeat steps 3 and 4 until n_{goal} is reached, then add it to P'.

Algorithm 4 Backward optimization method. It takes the solved path P after search, and results in a path P' which has less nodes than P.

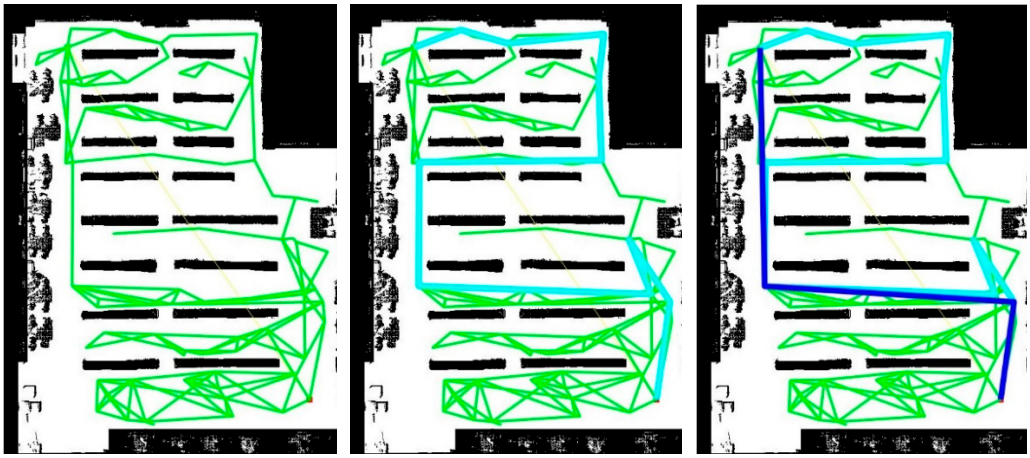
Input: Solved path P' in which is a set of nodes includes source n_{src} and goal n_{goal}.

```
P'.add(n_src);
n ← n_src;
n_m ← n + 1;
while n != n_goal do
    if P.find(n) > 1
        n ← get_final_index(n);
        n_m ← n + 1;
    else
        e_{n,m} ← EDGE(n, n_m);
        if CHECK_COLLISION (e_{n,m}) == COLLISION_FREE
            n_m ← n_{m+1};
        else
            P'.add_node(n_{m-1});
            n ← n_{m-1};
            n_m ← n + 1;
return P'
```

Figure 10. Pseudocode of backward path connection check.

4.2. Forward Path Connection Check

The forward connection check method also starts from the source node as the first determined node but differs from the backward connection check, in that it checks the visibility starting from the goal node and searching forward node by node to the determined node. The aim of the forward connection check is to cope with the backward connection check in the case where two distant visible nodes exist with invisible nodes in between. As shown in Figure 11b, the path (cyan lines) obtained by the backward connection check returns to have the path extended around the upper, right, and lower sides of the obstacle area at the top of the map. However, in reality, the path can be directly connected down the left side of the obstacle area. At this point, the original path can be optimized by the forward connection check; as shown in Figure 11c, the resulting path (blue lines) after the forward connection check can directly connect the two nodes on the left side of the above obstacle area, avoiding the redundant path nodes.



(a) Collision-free path before optimization **(b)** Backward connection checked path **(c)** Forward connection checked path

Figure 11. Path postprocessing optimization: (**a**) an collision-free path (green lines) is obtained as an initially output of path planning; (**b**) backward connection checked path (cyan lines) firstly reduces unnecessary visits to the same nodes; (**c**) forward connection checked path (blue lines) further reduces redundant nodes.

The pseudocode of the forward path connection check is shown in Figure 12. More specifically, the procedure entails the following steps:

(1) Initialize the optimal path P'' by adding n_{src}.
(2) Input the backward optimal path P', starting from n_{src} as the determined node n and n_{goal} as the displacement node n_m.
(3) Perform collision check on edge $e_{n,m}$. If it passes, add n_m to P'', update n to n_m, and reset n_m; otherwise, let n_m move forward.
(4) Repeat steps 3 and 4 until n_{goal} is reached, then add it to P''.

The path postprocessing optimization combines the backward and forward connection checks to combine their respective benefits. First, considering the decreasing possibility of node visibility as their distance increases, the backward connection check from a fixed node is more efficient than the forward connection check from the end of the path, allowing for a faster traversal of the initial path. Second, the backward connection check, in the first step, cannot handle the situation where two distant visible nodes have invisible nodes in between, whereas the forward connection check, in the second step, can remedy this deficiency, because it checks the visibility from the other direction. Furthermore, the forward connection check, on the basis of the backward connection checked path,

can further improve the path quality, avoiding the efficiency issues caused by directly performing it on the initial solved path after the path search.

The path postprocessing optimization method avoids multiple visits to the same node, minimizes unnecessary visits to redundant nodes, creates a straightening effect in the path shape, and reduces the final path length.

Algorithm 5 Forward optimization method. It takes the backward optimal path P' and results in a path P''.

Input: Backward optimal path P' including source n_{src} and goal n_{goal}.

P''.add(n_{src});
$n \leftarrow n_{src}$;
$n_m \leftarrow n_{goal}$;
while n != n_{goal} **do**
while $n_m > n$ **do**
$e_{n,m} \leftarrow$ EDGE(n, n_m);
if CHECK_COLLISION($e_{n,m}$) == COLLISION_FREE
P''.add_node(n_m);
$n \leftarrow n_m$;
$n_m \leftarrow n_{goal}$;
else
$n_m \leftarrow n_{m-1}$;
return P''

Figure 12. Pseudocode of path forward connection check.

5. Experimental Results

5.1. Source Data and Environment

The source data for the experiments in this study were 3D point cloud data of an indoor environment acquired using light detection and ranging (LiDAR) scanning equipment, including two indoor scenes whose details are listed in Table 1.

Table 1. Source data of indoor environment.

Name	Description	No. Points	Data Quality	Overview
Scene 1	Wuhan University library east reading room, 2nd Floor	2,848,055	locally vacant	Figure 14a
Scene 2	Underground parking lot	6,794,787	locally vacant	Figure 14b

The map data we used in the path planning experiments included two reduced-dimensional raster maps of the aforementioned indoor environment and two virtual binary image maps [63] used for comparison. The original point cloud data were missing some scans, which necessitated manual completion of the vacant areas to ensure the integrity of the maps before the subsequent operations. The details of the four indoor environment maps are listed in Table 2, and Figure 13 provides overviews.

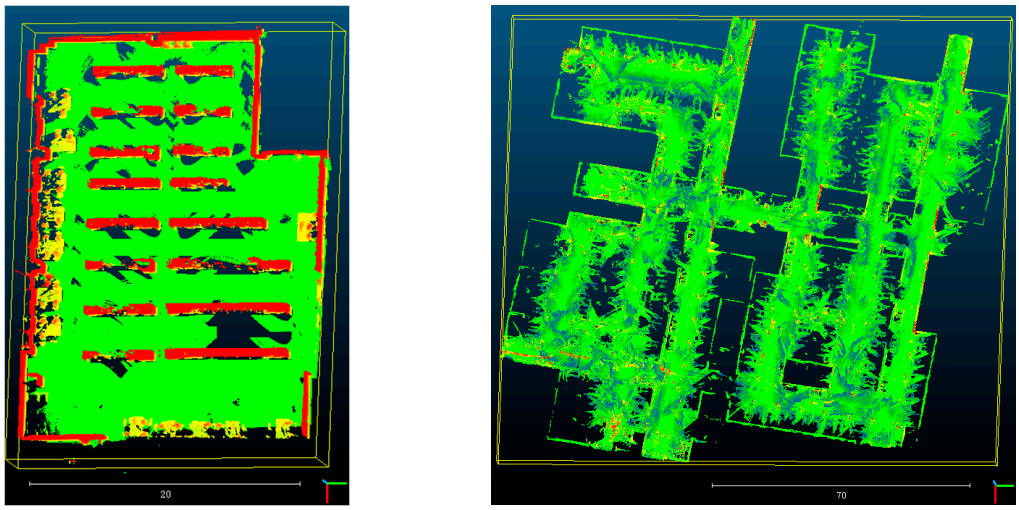
The proposed indoor environment modeling method involves the processing of point clouds, and we implemented the associated experiments using C/C++ programming with the Point Cloud Library (PCL) [64]. We also used the image processing library OpenCV to generate a reduced-dimensional raster map of the indoor environment.

The experimental simulation platform was MatLab, with an Intel® Core™ i7-7700HQ 2.80GHz CPU and 8GB RAM. We interpreted a reduced-dimensional raster map of the indoor environment as a two-dimensional simulation space and then implemented the path planning algorithm in the same space.

Table 2. Map data of the indoor environment.

Name	Description	Map Size	Data Quality
Map1 [1]	Binary image of simple obstacles	500 × 500 (px)	good
Map2	Binary image of a complex maze	500 × 500 (px)	good
Map_lib [2]	Reduced-dimensional raster map of Scene 1	575 × 773 (px)	locally vacant
Map_pkl	Reduced-dimensional raster map of Scene 2	682 × 625 (px)	locally vacant

[1] Binary images of Map1 and Map2 were obtained from web resources; [2] Map_lib and Map_pkl are reduced-dimensional raster maps generated using the method proposed in this study, where vacant areas in original data were filled, and external areas were set to black.



(a) Scene 1 **(b)** Scene 2

Figure 13. Indoor environment map for path planning (**a**–**d**).

To further simulate the condition of a typical UAV onboard system with limited resources, we validated and evaluated the performance of our methods using Manifold2-C, an onboard PC specially designed by DJI for their UAV products. The configuration was an Intel® Core™ i7-8550U 1.80GHz CPU with 8G RAM. We manually limited its CPU usage down to 30%; otherwise, we found that it had a faster computation speed than in the previous experimental environment, as the CPU and RAM on Manifold2-C are more up-to-date and offer better performance under the same input power.

5.2. Evaluation Metrics

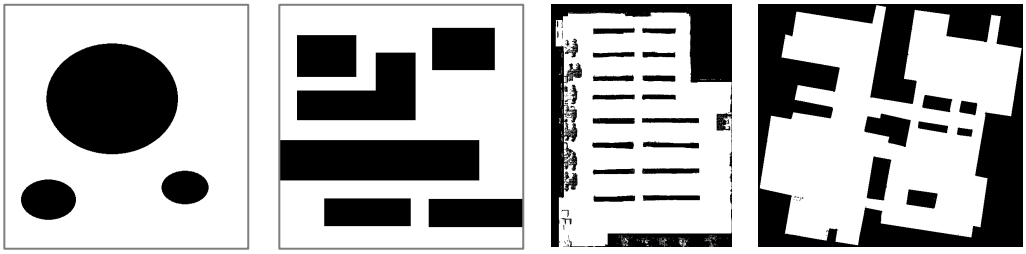
We evaluated the proposed methods using three metrics: pathfinding success rate, planning time, and path length.

Pathfinding success rate is the basic metric as it indicates the practicability of the path-planning method. If the improvement in the path-planning method results in a significant decrease in the pathfinding success rate compared with the original method, even if it achieves a considerable improvement in other aspects, such changes are meaningless because the algorithm no longer satisfies the most fundamental requirement of solving a path from the source to the goal.

We focused on planning time as a metric as some UAV autonomous flight applications involve collaboration between various onboard systems. If the path planning procedure is too slow, a series of subsequent operations will need to wait and will stagnate, which not only does not meet the real-time UAV positioning and planning requirements but is also detrimental to the safety of autonomous UAV flight.

Path length reflects the quality of the path as determined by the path planning method. The shorter the path length, the shorter the flight time, which can reduce unnecessary power

consumption and help the UAV avoid energy shortages when performing autonomous flight missions, thereby enhancing flight safety.



(a) Map1 **(b)** Map2 **(c)** Map_lib **(d)** Map_pkl

Figure 14. Point cloud overviews of indoor environment. We removed ceiling points from the point clouds for better observation (**a**,**b**).

Because our focus in this study was reducing the path planning time, it is worth noting that the algorithm tends to reach a solution faster when determining the path than when identifying the shortest path between the source and goal. On this basis, path postprocessing optimization is conducted to account for a non-shortest path length. As a result, the final path may be longer than the basic methods, but this is acceptable as long as the deviation is not excessive.

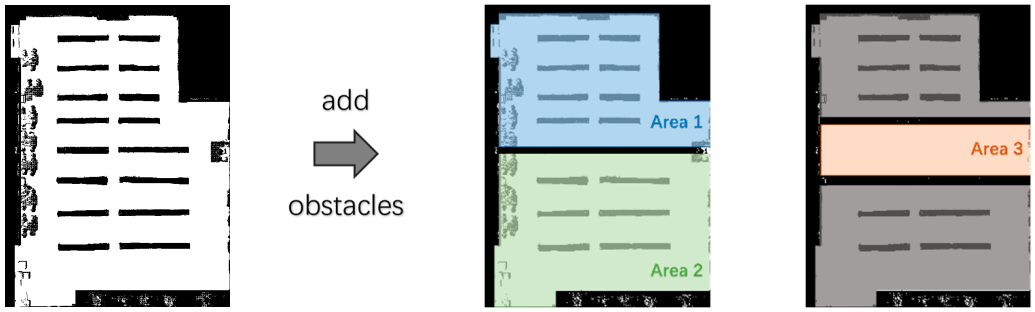
5.3. Experiments on Reduced-Dimensional Rasterization of Indoor Environment

In this study, we conducted experiments with real-world indoor scenes, including a library reading room and an underground parking lot. We performed the rasterization of the indoor environment based on the reduced-dimensional raster map generated by point cloud projection. The black grids represent impassable areas in the indoor environment, such as obstacles and boundaries, whereas the white grids represent passable and occupiable free space.

For simple environments, we projected the point cloud at the altitude midway between the floor and ceiling of the scene, within a 1 m height range. Moreover, we created a multilayer grid map to test our planning method in a more complex scenario where the distribution of obstacles varied among the different altitudes of the environment. We had no existing source data that met our needs, so we manually created a multilayer map by editing obstacles in the map that divided the maps into several areas. Different layers of the multilayer grid map are shown in Figure 15, representing the reduced-dimensional maps of the environment at different altitudes. We assumed that the altitude of Layer 1 was lower than that of Layer 2. On this particular map, if a UAV wanted to move from Area 1 to Area 2, it had to execute the following flight:

(1) Start from somewhere in Area 1 at the altitude of Layer 1.
(2) Move to the overlapping area of Areas 1 and 3.
(3) Ascend to the altitude of Layer 2.
(4) Move to the overlapping area of Areas 3 and 2.
(5) Descend to the altitude of Layer 1.
(6) Continue the flight in Area 2 at the altitude of Layer 1.

The map of the library reading room in Figure 13c shows that a portion of the obstacle areas representing the library cabinets was broken. This was due to missing scans in the original point cloud data, which we resolved by recollecting higher-quality indoor space point cloud data. The obstacles near the wall were the projections of desks and chairs. Although some space between them and the ceiling in the actual indoor space was free, this kind of space accounted for a small portion of the total space and had no substantial impact on the connectivity of the free space, so the waste of this portion of the space was still acceptable.



(a) Original. **(b)** Layer 1. **(c)** Layer 2.

Figure 15. Multilayer grid maps. Based on original Map_lib (**a**), we manually added an obstacle wall in the middle of Layer 1 (**b**), and divided the map into Areas 1 and 2; two obstacle walls in Layer 2 (**c**) near the aforementioned wall and enclosed Area 3.

The map of the underground parking lot in Figure 13d shows that vehicles in the parking lot did not notably interfere with the point cloud projection because we specified the projection method that extracts the height of the upper middle region of the parking lot.

Using the reduced-dimensional rasterization of the indoor environment considerably simplified the environment and met the data requirements of the subsequent path-planning experiments. Table 3 lists a comparison of the data before and after modeling of the two indoor scenes in this experiment, demonstrating that the reduced-dimensional modeling substantially reduced the data volume. The final maps generated by the method proposed in this study were essentially images, and their spatial accuracy was freely adjustable, and the corresponding data size changed with the image resolution.

Table 3. Data comparison before and after modeling.

Scene			Description	Format	Resolution	Size
Library reading room	Before		Point cloud	.pcd	2,848,055 (pts)	119 MB
	After	Small	Raster map	.bmp	575 × 773 (px)	435 KB
		Medium			1150 × 1546 (px)	1.74 MB
		Large			2300 × 3092 (px)	6.96 MB
Under-ground parking lot	Before		Point cloud	.pcd	6,794,787 (pts)	155 MB
	After	Small	Raster map	.bmp	682 × 625 (px)	418 KB
		Medium			1363 × 1250 (px)	1.62 MB
		Large			2724 × 2498 (px)	6.49 MB

5.4. Experiments on Network Construction Based on Connection Distance

The purpose of setting the connection distance parameter in network construction is to avoid collision checks between distant nodes, because connections that span a larger area are more likely to intersect with obstacles in the environment.

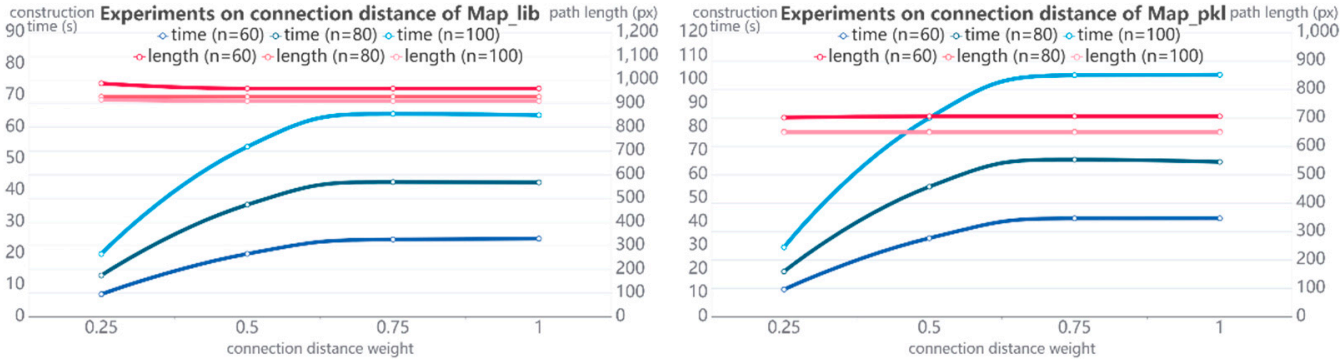
In the experiment, we defined the connection distance *c_dis* as follows:

$$c\_dis = w_{cd} * \sqrt{width_{map}^2 + height_{map}^2}, \tag{8}$$

where $width_{map}$ and $height_{map}$ denote the size of the grid map, and w_{cd} denotes the scale factor, i.e., connection distance weight.

Using Map_lib and Map_pkl data, we separately set w_{cd} to 0.25, 0.5, 0.75, and 1, i.e., the connection distance was 1/4, 1/2, 3/4, and 1 times the length of the map diagonal, respectively. For different numbers of nodes, we recorded the number of connected edges and network construction time, as well as the path length obtained by path search based on this network. The experimental results are listed in Table A1.

The network construction time and path length for the connection distance experiments are shown in Figure 16. With smaller connection distances (w_{cd} = 0.25), the number of constructed network edges was smaller and the construction time was shorter, but this resulted in a less successful path search and a longer path.



(a) Map_lib **(b)** Map_pkl

Figure 16. Construction time and path length of Map_lib (**a**) and Map_pkl (**b**).

As the connection distance weight increased, the network construction time accordingly increased, and the path length obtained from the path search tended to be shorter as a result of the increased number of possible connecting edges. However, the growth in the number of collision-free edges and the decrease in path length tended to be flat, and the number of colliding edges markedly increased. These redundant checks also resulted in a larger waste of network construction time, confirming the hypothesis that increasing distance decreases the possibility of collision-free edges. In the process of network construction, the algorithm performance could be enhanced by selecting a moderate connection distance.

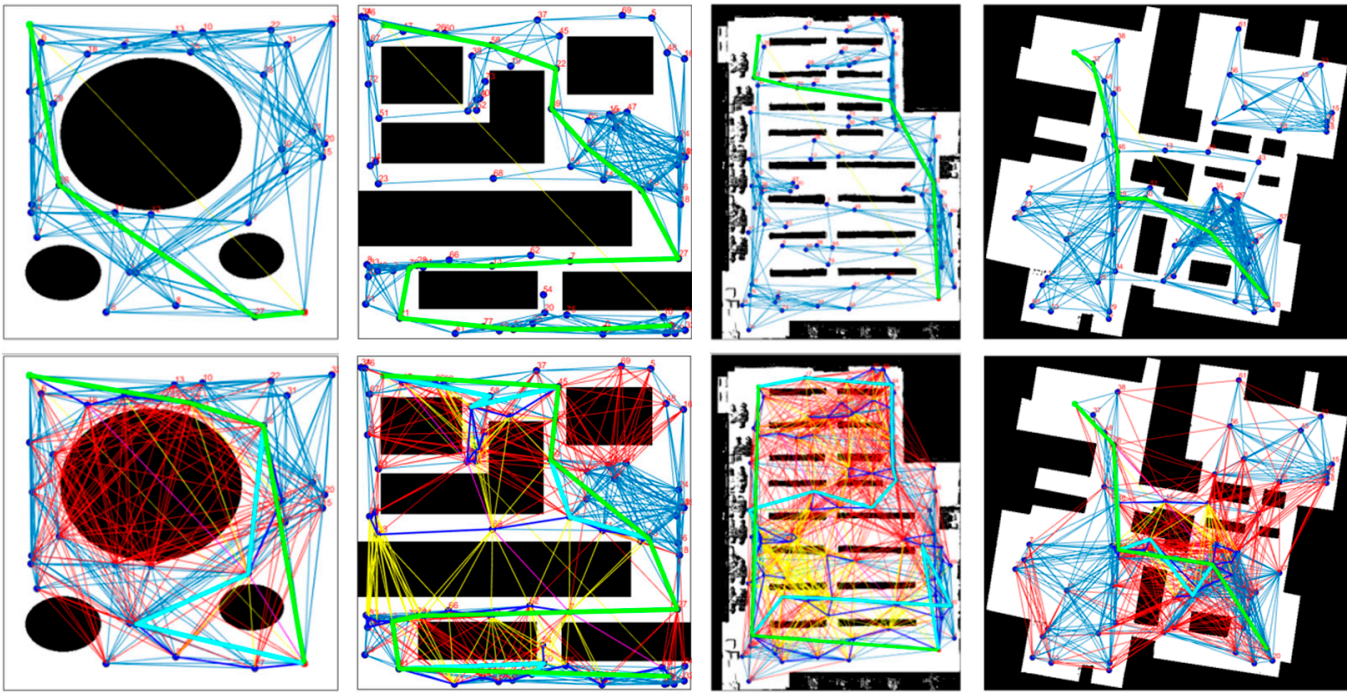
5.5. Experiments on Improved Probabilistic Roadmap Planning

In this study, we used the four raster maps shown in Figure 13 to conduct probabilistic roadmap planning experiments. For each map, we positioned the source and goal near the diagonal position of the map, varied the number of nodes and connection distance, and repeated the experiment to record the pathfinding success rate, planning time, and path length of the basic and improved PRM methods.

The path planning results of the four maps are shown in Figure 17, where blue dots represent network nodes; blue and red lines represent visible and nonvisible edges, respectively; yellow lines represent updated edges during path search; cyan lines represent the forward-checked optimal path; and green lines represent the backward-checked optimal path, i.e., the final path result.

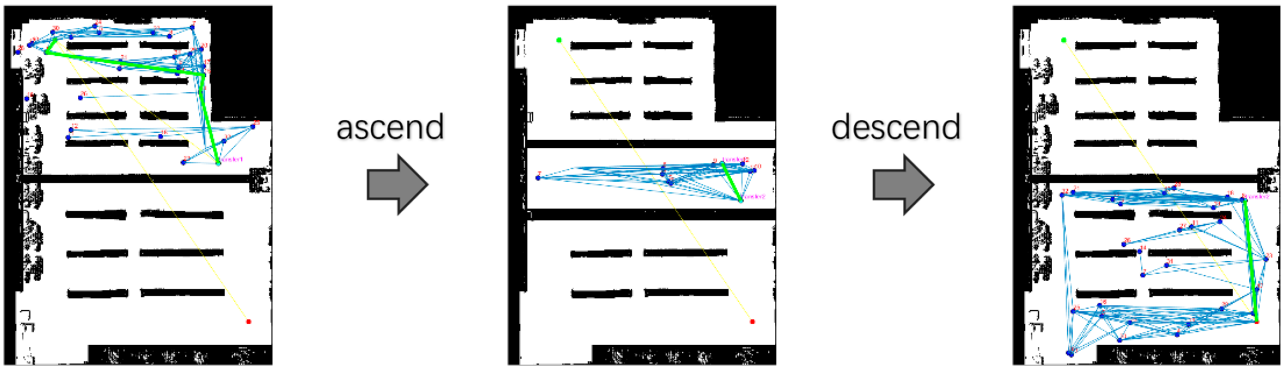
Comparing the result of the basic PRM with that of the improved PRM, we found two common cases. In the first case, the majority of nodes in both paths were the same but had slightly different path shapes; in the second case, the two paths had considerably different routes because we adopted an incremental update strategy and the path search followed the rule of greedy extension rather than that of global length shortening. Notably, as a result of our path optimization, the path length of our method was often comparable to that of the basic one, despite the route difference.

The path planning results of the multilayer grid map are shown in Figure 18. The source and the goal were the same as in Map_lib, with the source located in Area 1 and the goal in Area 2 (as shown in Figure 15). To travel from start to end, the UAV adjusted its flight altitude, i.e., ascending in the overlapping of Areas 1 and 3 and descending in the overlapping of Areas 3 and 2. The complete path from start to end was composed of three subpaths connected by two transfer nodes where the UAV performed vertical movements.



(a) $n = 30$, $cd = 0.5$. **(b)** $n = 75$, $cd = 0.25$. **(c)** $n = 80$, $cd = 0.25$. **(d)** $n = 60$, $cd = 0.25$.

Figure 17. Path planning results of grid maps of node number (n) and connection distance (cd). The first row is the results of basic PRM, and the second row is the results of improved PRM. The columns from left to right are Map1 (**a**), Map2 (**b**), Map_lib (**c**), and Map_pkl (**d**).

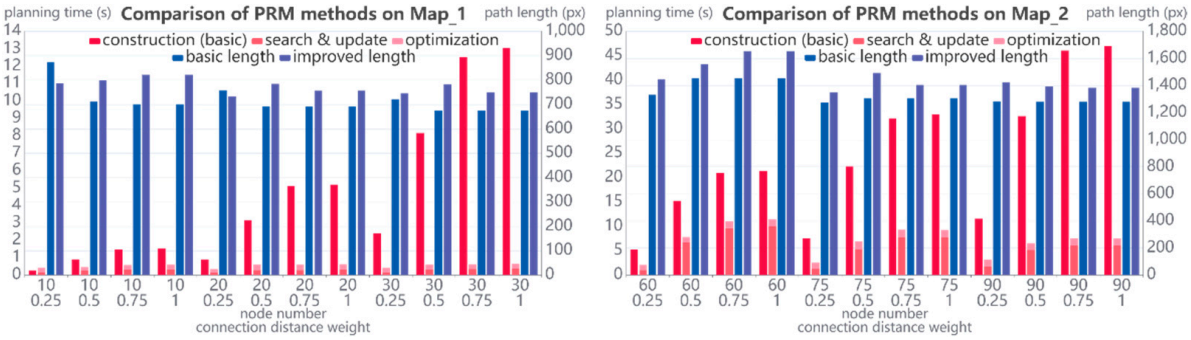


(a) Subpath in Area 1 **(b)** Subpath in Area 3 **(c)** Subpath in Area 2

Figure 18. Path planning result of multilayer grid map: three subpaths are searched in Area 1 (**a**), Area 3 (**b**), and Area 2 (**c**), respectively, and the subpaths are connected by two transfer nodes in the overlapping areas.

We repeated the experiments by configuring different connection distances for a different number of nodes. We recorded the planning time and path length of the basic and improved PRM methods. The comparative results of Map1 and Map2 are listed in Table A2. The basic PRM spent most of its time constructing the network, whereas the improved PRM spent most of its time on path search, network update, and path postprocessing optimization, as the check of network connectivity was deferred.

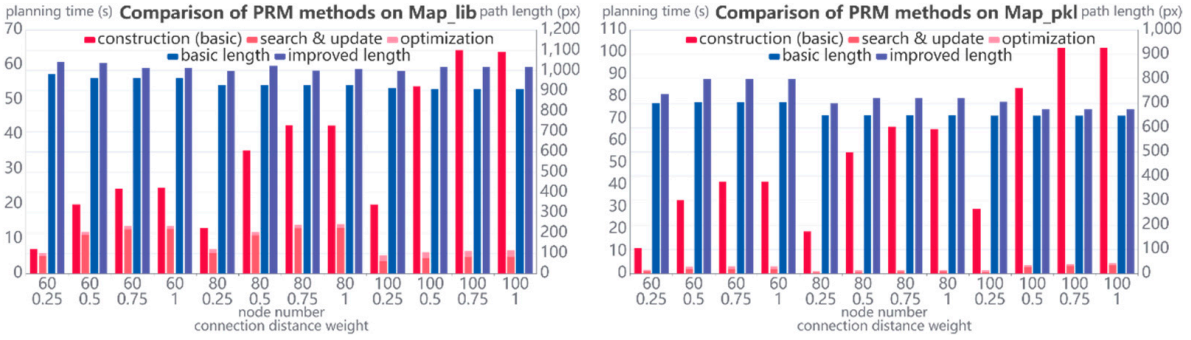
For Map1 and Map2, we calculated the average planning time and average path length ratio before and after the improvement, as shown in Figure 19. The planning time of the improved PRM was only approximately 30% of that of the basic PRM, resulting in a substantial reduction in path planning time. Due to some strategies adopted by the algorithm, it tended to find a path as quickly as possible rather than determine the shortest path between the source and the goal; consequently, the path length was longer than that of the basic PRM by less than 10%, which is acceptable given the considerable increase in planning time. In addition, the planning time and path length showed that as the number of nodes increased, the advantage in the planning time of the proposed method became more apparent.



(a) Map1 **(b)** Map2

Figure 19. Planning time and path length of Map1 (**a**) and Map2 (**b**).

We also conducted the same experiments for two reduced dimensional raster maps of indoor environments, Map_lib and Map_pkl; the results are listed in Table A3. From a comparison of the results of the basic and improved PRM on Map_lib and Map_pkl, as shown in Figure 20, our conclusions were basically the same: the improved PRM provided an advantage over the basic PRM in terms of planning time, at the cost of an increase in path length, which was acceptable.



(a) Map_lib **(b)** Map_pkl

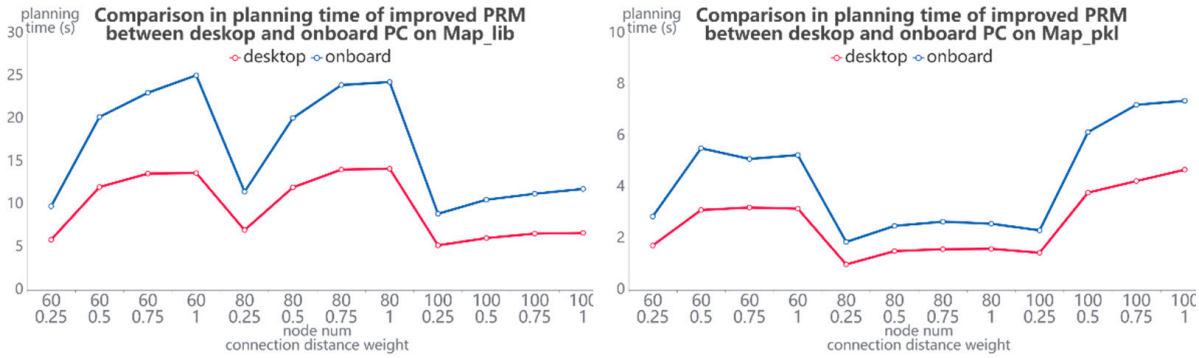
Figure 20. Planning time and path length of Map_lib (**a**) and Map_pkl (**b**).

Comparing the binary image map and the reduced-dimensional raster map of the indoor environment, we found that for the simple maps (Map1 and Map_pkl), the planning time stability in the improved PRM was high, i.e., the planning time did not notably fluctuate with changes in the number of nodes or the connection distance. For the complex maps (Map2 and Map_lib), the stability decreased, but was still considerably better than that of the basic PRM.

We also conducted experiments on an onboard PC to simulate the integration of our planning method into an autonomous UAV system. We tested our improved PRM method

on the onboard PC and evaluated the planning time using the same input data as the original experiments. At first, we did not set a resource limit on the onboard PC, but due to its hardware configuration being more up-to-date than that of our desktop PC, it performed even better. Therefore, we manually limited the CPU usage down to 30% to more accurately simulate a resource-limited configuration.

The experimental results of the improved PRM on Map_lib and Map_pkl on the onboard PC (with CPU usage limited to 30%) are listed in Table A4, and they are shown in Figure 21. The planning times in both experimental environments exhibited comparable trends under the same input data. Additionally, the consumed time was fundamentally influenced by the hardware configuration, whereas our planning method maintained appropriate performance despite resource limitations. Moreover, we showed that the current onboard hardware was capable of high-level configurations (except for those dedicated to micro vehicles), so scholars can more easily deploy and test their methods on onboard computing platforms with high performance.



(a) Map_lib **(b)** Map_pkl

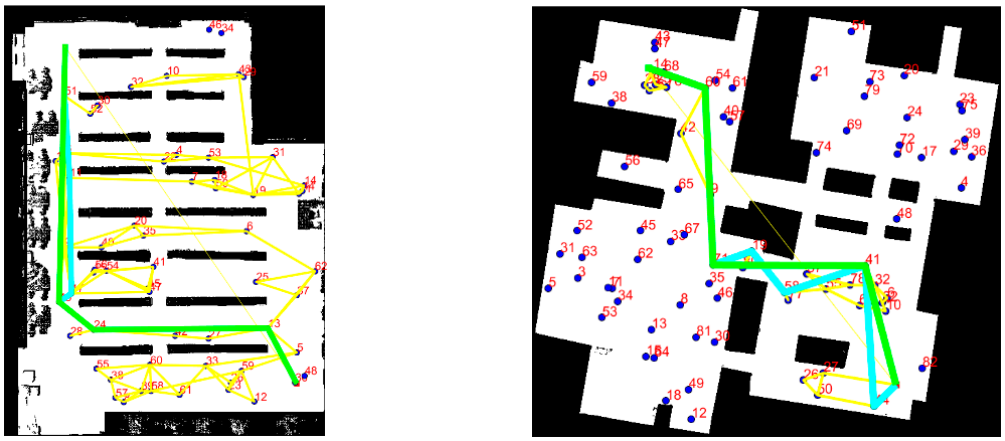
Figure 21. Planning time of Map_lib (**a**) and Map_pkl (**b**), with comparison between desktop and onboard PC configurations.

5.6. Experiments on Path Postprocessing Optimization

Using the same reduced-dimensional raster maps of the indoor environment, we conducted the experiments of path post-processing optimization. Experimental results are shown in Figure 22, where thick yellow lines represent updated edges, yellow lines represent collision-free path after path search, cyan lines represent the backward-checked optimal path, and green line represents the forward-checked optimal path, i.e., the final path result.

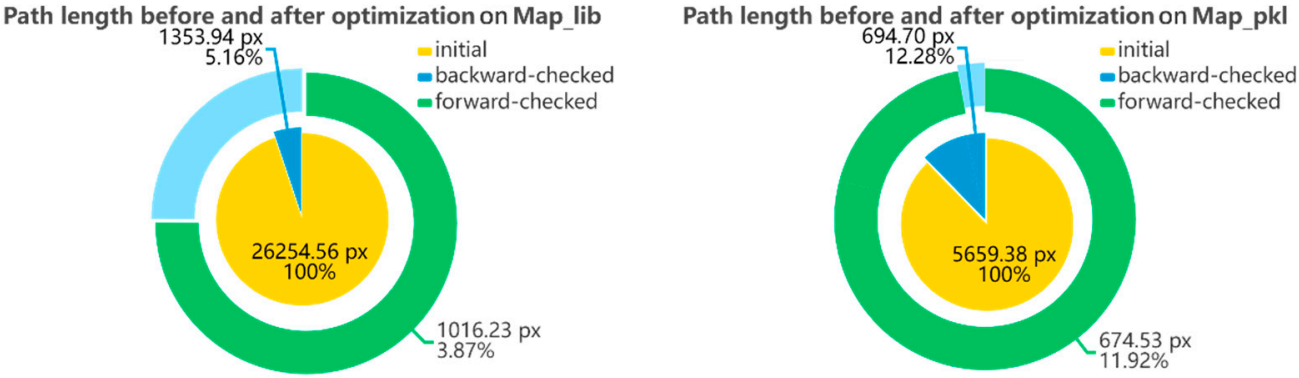
In Map_lib and Map_pkl, the paths were solved using the basic and improved PRM methods for comparison. The initially obtained collision-free paths were then optimized using two-step path postprocessing. The experimental results are listed in Table A5.

From the path postprocessing optimization results shown in Figure 23, we found that the proposed optimization method substantially improves the path quality for the initially obtained paths, with longer lengths and higher path repetition rates in the improved PRM, as the method minimized redundant nodes and allowed the path to attain a "straightened" route. This path postprocessing optimization method solves the problem where the initial path quality obtained by the improved PRM method is inferior to that of the basic one, completing the proposed path planning scheme. As a result, the optimization method avoids redundant motions and ensures a collision-free and direct path for the UAV, which is conducive to the safety of autonomous flight in indoor environments.



(a) Map_lib **(b)** Map_pkl

Figure 22. Path postprocessing optimization result of Map_lib (**a**) and Map_pkl (**b**).



(a) Map_lib **(b)** Map_pkl

Figure 23. Path length before and after optimization on Map_lib (**a**) and Map_pkl (**b**).

6. Discussion

In this study, we primarily focused on modeling an indoor environment and improving the PRM path-planning method. We assumed the UAV was a quadrotor model and reduced the controls to 3D translation and rotation in yaw angle. This assumption is relatively simple but was sufficient for the purposes of this study, as numerous flight control products and software have already encapsulated and integrated UAV operations into simple commands. More in-depth research can be conducted on the topic of UAV control.

The data source for indoor environment modeling was a point cloud. Furthermore, we used a point cloud projection scheme to generate reduced-dimensional raster maps of indoor environments. Although this method preserves the semantic understanding of indoor environment elements and the modeling speed is faster, the problem of noise points present in the point cloud must still be solved, as determining whether these noise points represent tiny obstacles or actually are noise points is crucial to the safety of indoor UAV flights. In the actual process of map generation, we must also be problem-specific because the presence of too many noise points will impede the efficiency of solving the path, and eliminating these noise points may pose a safety risk, so we should strike a balance between these two factors.

Considering the complexity of indoor environments, we generated grid maps at various altitudes and constructed a multilayer map. This method increases the robustness of path search and the applicability of the UAV.

The proposed improved PRM planning method aims to solve the computational efficiency problem caused by the basic PRM while minimizing the impact on the path length. The results of our experiments showed that the proposed method is effective. However, the efficiency of the proposed method may decrease in overly complex indoor situations, because if collision checks on edges are not performed when constructing the original network, a large number of edge adjacency information updates are required in the subsequent search and update process; each update necessitates at least one local path search, resulting in a substantial decrease in efficiency. To avoid repeated checks, it is recommended to check the connection between all nodes at the beginning.

The key to our improvement provided by our strategy is reducing a large number of collision checks in the basic method, because collision checks performed during network construction may be meaningless for two reasons: First, as the distance between nodes increases, the possibility of visibility decreases, because more obstacles created blockages between them. Second, the farther nodes are from the source and goal nodes, the less likely they are to become candidates, so checks on them are often unnecessary in the end. As a result, we adopted the strategy of constructing first and checking later to eliminate as many unnecessary checks as possible and searched for paths based on the overall effectiveness of the candidate nodes from high to low, which can more quickly solve the paths while retaining the ability to search for all nodes because, in extreme cases after traversing through all nodes in the space, as long as a path solution exists, the path solution should be obtained.

7. Conclusions

In this study, we developed an improved probabilistic roadmap planning method for safe indoor UAV flight, under the assumption of a quadrotor UAV.

We modeled the indoor environment with point cloud projection and represented it with reduced-dimensional raster maps. The grid map model performed well in terms of environment representativity and modeling efficiency. Furthermore, our proposed multilayer gird map model, an innovation over the original single-layer model, contributes to improving path planning effectiveness, particularly in complex environments.

Based on the grid map model, we conducted path planning experiments and improved upon the basic PRM planning method in terms of network construction, search strategy, and path optimization. Our method remarkably outperformed the basic PRM in computational efficiency while maintaining a reasonable path length. It also showed high-quality performance on both desktop PCs and resource-limited onboard platforms, laying the foundation for indoor UAV applications and fulfilling the requirements for autonomous UAV flight safety.

Author Contributions: Conceptualization, Q.H. and S.W.; methodology, Q.J. and P.Z.; software, Q.J. and P.Z.; validation, Q.H., S.W. and M.A.; formal analysis, Q.H. and S.W.; investigation, Q.J., P.Z. and M.A.; resources, P.Z. and M.A.; data curation, Q.J., P.Z. and M.A.; writing—original draft preparation, Q.J.; writing—review and editing, Q.H. and S.W.; visualization, Q.J.; supervision, Q.H. and S.W.; project administration, Q.H.; funding acquisition, Q.H. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the National Natural Science Foundation of China (NSFC), grant number 42101441, and the Fundamental Research Funds for the Central Universities.

Data Availability Statement: Data and source codes in this study will be made available on https://github.com/Jin-qg/iPRM before 1 March 2023.

Conflicts of Interest: The authors declare no conflict of interest. The funders had no role in the design of the study; in the collection, analyses or interpretation of data; in the writing of the manuscript, or in the decision to publish the results.

Appendix A

Table A1. Path planning results.

Node Num.	1	2	Edge Num. 3	4	5	6	Path Length
			Map_lib				
60	0.25	80%	213.38	417.50	1260.13	7.1084	984.82
60	0.5	100%	293.30	1235.40	362.30	19.8989	963.19
60	0.75	100%	297.30	1580.00	13.70	24.4590	963.19
60	1	100%	297.30	1593.70	0.00	24.7062	963.19
80	0.25	80%	380.88	778.13	2162.00	13.0713	928.56
80	0.5	100%	495.60	2272.50	552.90	35.4468	928.21
80	0.75	100%	501.50	2807.80	11.70	42.6125	928.21
80	1	100%	501.50	2819.50	0.00	42.5122	928.21
100	0.25	100%	548.90	1246.80	3355.30	19.8673	915.75
100	0.5	100%	710.60	3534.80	905.60	53.7906	910.33
100	0.75	100%	718.50	4411.20	21.30	64.2075	910.33
100	1	100%	718.50	4432.50	0.00	63.7934	910.33
			Map_pkl				
60	0.25	90%	400.11	264.44	1226.44	11.4924	700.32
60	0.5	100%	476.50	1143.50	271.00	33.1575	704.85
60	0.75	100%	482.20	1407.90	0.90	41.4939	704.85
60	1	100%	482.20	1408.80	0.00	41.5201	704.85
80	0.25	100%	697.90	486.30	2136.80	19.0639	650.89
80	0.5	100%	866.40	2020.80	433.80	54.8623	650.74
80	0.75	100%	878.40	2440.90	1.70	66.3098	650.74
80	1	100%	878.40	2442.60	0.00	65.3203	650.74
100	0.25	100%	1050.80	747.60	3352.60	29.3165	648.71
100	0.5	100%	1264.70	3164.30	722.00	83.9546	648.69
100	0.75	100%	1280.40	3868.50	2.10	101.9095	648.69
100	1	100%	1280.40	3870.60	0.00	102.1078	648.69

1 Connection distance weight; 2 Path finding success rate; 3 Edge number of collision-free edges; 4 Edge number of collided edges; 5 Edge number of skipped; 6 Network construction time.

Table A2. PRM comparative results of Map1 and Map2.

Node Num.	1	2	Time of Improved PRM 3	4	5	Total	Path Length Basic	Improved
				Map1				
10	0.25	0.2655	0.1570	0.2500	0.4070	873.69	786.16	
10	0.5	0.9056	0.2551	0.1994	0.4546	711.68	797.93	
10	0.75	1.4807	0.3016	0.2715	0.5731	700.87	821.82	
10	1	1.5241	0.3183	0.2689	0.5872	700.87	821.82	
20	0.25	0.8855	0.1327	0.2073	0.3400	757.33	732.86	
20	0.5	3.1325	0.2857	0.2985	0.5842	692.63	784.42	
20	0.75	5.1130	0.2876	0.3155	0.6031	691.70	756.36	
20	1	5.1925	0.3107	0.3138	0.6245	691.70	756.46	
30	0.25	2.3752	0.1563	0.2483	0.4046	721.99	746.97	
30	0.5	8.1504	0.3155	0.2887	0.6042	675.18	783.13	
30	0.75	12.5140	0.3517	0.2684	0.6201	674.90	748.98	
30	1	13.0509	0.3703	0.2669	0.6372	674.90	748.98	
Average time ratio			33.0%		Average length ratio		108.9%	
				Map2				
60	0.25	5.2805	1.0070	1.0493	2.0563	1330.90	1443.88	
60	0.5	15.2353	6.6992	1.1010	7.8002	1454.97	1556.92	
60	0.75	20.9249	9.6638	1.3777	11.042	1454.71	1651.95	
60	1	21.3038	10.019	1.3912	11.410	1454.71	1651.95	
75	0.25	7.5534	1.2740	1.2716	2.5456	1274.06	1348.04	

Table A2. *Cont.*

Node Num.	1	2	Time of Improved PRM			Path Length	
		3	4	5	Total	Basic	Improved
75	0.5	22.2787	5.3312	1.5318	6.8630	1306.63	1493.31
75	0.75	32.1494	7.7480	1.5358	9.2838	1306.61	1404.32
75	1	33.0011	7.7118	1.5358	9.2477	1306.61	1404.32
90	0.25	11.5591	1.7538	1.3451	3.0989	1281.55	1422.85
90	0.5	32.6026	5.1573	1.3882	6.5456	1279.15	1391.26
90	0.75	46.0452	6.1180	1.4262	7.5442	1279.09	1382.75
90	1	46.9495	6.0681	1.4318	7.4999	1279.09	1382.75
Average time ratio			33.1%		Average length ratio		109.5%

1 Connection distance weight; 2 Time of basic PRM; 3 Network construction; 4 Update and search; 5 Path optimization.

Table A3. PRM comparative results of Map_lib and Map_pkl.

Node Num.	1	2	Time of Improved PRM			Path Length	
		3	4	5	6	Total	7	8
				Map1				
60	0.25	7.11	0.00	5.0644	0.8180	5.8843	984.82	1043.95
60	0.5	19.90	0.01	11.1111	0.9270	12.0476	963.19	1037.95
60	0.75	24.46	0.00	12.7196	0.9000	13.6243	963.19	1014.45
60	1	24.71	0.00	12.7823	0.9002	13.6872	963.19	1014.45
80	0.25	13.07	0.01	5.9111	1.0935	7.0123	928.56	997.52
80	0.5	35.45	0.01	11.0135	0.9830	12.0074	928.21	1024.94
80	0.75	42.61	0.01	13.0456	1.0274	14.0872	928.21	1000.56
80	1	42.51	0.01	13.1091	1.0667	14.1884	928.21	1007.73
100	0.25	19.87	0.01	3.6025	1.6199	5.2286	915.75	998.80
100	0.5	53.79	0.01	4.4192	1.6485	6.0815	910.33	1018.50
100	0.75	64.21	0.02	4.8817	1.7028	6.6048	910.33	1017.93
100	1	63.79	0.02	4.9162	1.7433	6.6801	910.33	1017.93
Average time ratio			28.5%		Average length ratio		108.5%	
				Map2				
60	0.25	11.49	0.00	1.0242	0.7097	1.7357	700.32	738.45
60	0.5	33.16	0.01	2.0315	1.0820	3.1212	704.85	799.88
60	0.75	41.49	0.01	2.1125	1.0986	3.2173	704.85	799.88
60	1	41.52	0.00	2.0853	1.0827	3.1727	704.85	799.88
80	0.25	19.06	0.01	0.4200	0.5734	0.9996	650.89	698.75
80	0.5	54.86	0.01	0.8500	0.6561	1.5171	650.74	721.73
80	0.75	66.31	0.01	0.8879	0.6932	1.5919	650.74	721.55
80	1	65.32	0.01	0.9127	0.6798	1.6064	650.74	721.55
100	0.25	29.32	0.01	0.6826	0.7624	1.4544	648.71	705.78
100	0.5	83.95	0.02	3.0867	0.6923	3.7976	648.69	674.53
100	0.75	101.91	0.02	3.5807	0.6483	4.2493	648.69	674.53
100	1	102.11	0.02	4.0346	0.6420	4.6936	648.69	674.53
Average time ratio			4.8%		Average length ratio		109.0%	

1 Connection distance weight; 2 Time of basic PRM; 3, 4 Network construction time; 5 Update and search; 6 Path optimization; 7 Basic.; 8 Improved.

Table A4. Improved PRM results on onboard PC (with CPU usage limited to 30%).

Node Num.	1	Planning Time			
		2	3	4	Total
			Map_lib		
60	0.25	0.00	8.4516	1.3514	9.8063
60	0.5	0.01	18.6095	1.6332	20.2484
60	0.75	0.01	21.7281	1.3381	23.0730
60	1	0.01	23.4918	1.6114	25.1119
80	0.25	0.01	9.7605	1.7316	11.4980

Table A4. *Cont.*

Node Num.	1	Planning Time			
		2	3	4	Total
80	0.5	0.01	18.7435	1.3743	20.1281
80	0.75	0.01	22.3054	1.6590	23.9764
80	1	0.01	22.6307	1.6870	24.3299
100	0.25	0.01	6.2032	2.7115	8.9240
100	0.5	0.02	7.8199	2.7303	10.5656
100	0.75	0.02	8.5419	2.7033	11.2639
100	1	0.20	8.7333	2.8929	11.8291
Map_pkl					
60	0.25	0.00	1.8963	0.9678	2.8676
60	0.5	0.01	3.6753	1.8464	5.5278
60	0.75	0.01	3.4006	1.7047	5.1124
60	1	0.01	3.6033	1.6536	5.2637
80	0.25	0.01	0.7882	1.0823	1.8764
80	0.5	0.01	1.4127	1.0773	2.5008
80	0.75	0.01	1.5038	1.1492	2.6650
80	1	0.01	1.4970	1.0774	2.5867
100	0.25	0.07	1.0735	1.1801	2.3278
100	0.5	0.08	5.1668	0.9108	6.1574
100	0.75	0.02	6.3549	0.8481	7.2209
100	1	0.02	6.3881	0.9770	7.3810

1 Connection distance weight; 2 Network construction; 3 Update and search; 4 Path optimization.

Table A5. Path postprocessing optimization results.

Node Num.	1	Path Length of Basic PRM			Path Length of Improved PRM		
		Initial	2	3	Initial	4	5
Map_lib							
60	0.25	984.82	981.51	981.47	19,788.58	1347.54	1043.95
60	0.5	963.19	963.19	963.19	34,983.89	1325.51	1037.95
60	0.75	963.19	963.19	963.19	38,275.40	1259.18	1014.45
60	1	963.19	963.19	963.19	38,361.68	1259.18	1014.45
80	0.25	928.56	927.74	927.74	20,592.44	1263.47	997.52
80	0.5	928.21	928.21	928.21	33,226.28	1210.95	1024.94
80	0.75	928.21	928.21	928.21	37,746.75	1258.81	1000.56
80	1	928.21	928.21	928.21	38,090.94	1330.02	1007.73
100	0.25	915.75	914.77	914.77	11,689.17	1594.60	998.80
100	0.5	910.33	910.33	910.33	13,224.70	1436.51	1018.50
100	0.75	910.33	910.33	910.33	14,401.92	1447.90	1017.93
100	1	910.33	910.33	910.33	14,672.96	1513.58	1017.93
Avg. length ratio		100%	99.95%	100.00%	100%	5.16%	75.06%
Map_pkl							
60	0.25	700.32	700.30	700.30	3525.15	809.84	738.45
60	0.5	704.85	704.85	704.85	5544.48	894.79	799.88
60	0.75	704.85	704.85	704.85	5544.48	894.79	799.88
60	1	704.85	704.85	704.85	5544.48	894.79	799.88
80	0.25	650.89	650.74	650.74	1683.02	712.62	698.75
80	0.5	650.74	650.74	650.74	2729.33	733.67	721.73
80	0.75	650.74	650.74	650.74	2729.37	733.49	721.55
80	1	650.74	650.74	650.74	2729.37	733.49	721.55
100	0.25	648.71	648.69	648.69	2079.39	726.05	705.78
100	0.5	648.69	648.69	648.69	5480.68	694.70	674.53
100	0.75	648.69	648.69	648.69	5659.38	694.70	674.53
100	1	648.69	648.69	648.69	5659.38	694.70	674.53
Avg. length ratio		100%	100.00%	100.00%	100%	12.28%	97.10%

1 Connection distance weight; 2, 4 Backward-checked; 3, 5 Forward-checked.

References

1. Tsouros, D.C.; Bibi, S.; Sarigiannidis, P.G. A review on UAV-based applications for precision agriculture. *Information* **2019**, *10*, 349. [CrossRef]
2. Apolo-Apolo, O.; Martínez-Guanter, J.; Egea, G.; Raja, P.; Pérez-Ruiz, M. Deep learning techniques for estimation of the yield and size of citrus fruits using a UAV. *Eur. J. Agron.* **2020**, *115*, 126030. [CrossRef]
3. Carroll, S.; Satme, J.; Alkharusi, S.; Vitzilaios, N.; Downey, A.; Rizos, D. Drone-based vibration monitoring and assessment of structures. *Appl. Sci.* **2021**, *11*, 8560. [CrossRef]
4. Ren, H.; Zhao, Y.; Xiao, W.; Hu, Z. A review of UAV monitoring in mining areas: Current status and future perspectives. *Int. J. Coal Sci. Technol.* **2019**, *6*, 320–333. [CrossRef]
5. Sharma, V.; You, I.; Pau, G.; Collotta, M.; Lim, J.D.; Kim, J.N. LoRaWAN-based energy-efficient surveillance by drones for intelligent transportation systems. *Energies* **2018**, *11*, 573. [CrossRef]
6. Gupta, A.; Afrin, T.; Scully, E.; Yodo, N. Advances of UAVs toward future transportation: The State-of-the-Art, challenges, and Opportunities. *Future Transp.* **2021**, *1*, 326–350. [CrossRef]
7. Huang, H.; Savkin, A.V.; Huang, C. Scheduling of a parcel delivery system consisting of an aerial drone interacting with public transportation vehicles. *Sensors* **2020**, *20*, 2045. [CrossRef]
8. Ghelichi, Z.; Gentili, M.; Mirchandani, P.B. Logistics for a fleet of drones for medical item delivery: A case study for Louisville, KY. *Comput. Oper. Res.* **2021**, *135*, 105443. [CrossRef]
9. Pensieri, M.G.; Garau, M.; Barone, P.M. Drones as an integral part of remote sensing technologies to help missing people. *Drones* **2020**, *4*, 15. [CrossRef]
10. Liu, C.; Szirányi, T. Real-time human detection and gesture recognition for on-board uav rescue. *Sensors* **2021**, *21*, 2180. [CrossRef]
11. Jeelani, I.; Gheisari, M. Safety challenges of UAV integration in construction: Conceptual analysis and future research roadmap. *Saf. Sci.* **2021**, *144*, 105473. [CrossRef]
12. Maity, R.; Mishra, R.; Pattnaik, P.K. Flying robot path planning techniques and its trends. *Mater. Today Proc.* 2021; *in press*. [CrossRef]
13. Zhang, X.; Li, X.; Wang, K.; Lu, Y. *A Survey of Modelling and Identification of Quadrotor Robot*; Abstract and Applied Analysis; Hindawi: London, UK, 2014.
14. Meier, L.; Tanskanen, P.; Heng, L.; Lee, G.H.; Fraundorfer, F.; Pollefeys, M. PIXHAWK: A micro aerial vehicle design for autonomous flight using onboard computer vision. *Auton. Robot.* **2012**, *33*, 21–39. [CrossRef]
15. Quadcopters & Multirotor Drones—Draganfly—Driving Innovatin of Drones. Available online: https://draganfly.com/products/quadcopters-multirotors/ (accessed on 8 November 2022).
16. Matrice 300 RTK—Built Tough. Works Smart—DJI. Available online: https://www.dji.com/matrice-300 (accessed on 8 November 2022).
17. Wu, X.; Xiao, B.; Qu, Y. Modeling and sliding mode-based attitude tracking control of a quadrotor UAV with time-varying mass. *ISA Trans.* **2019**, *124*, 436–443. [CrossRef]
18. Jeon, H.; Song, J.; Lee, H.; Eun, Y. Modeling quadrotor dynamics in a wind field. *IEEE/ASME Trans. Mechatron.* **2020**, *26*, 1401–1411. [CrossRef]
19. Ji, D.; Wang, R.; Zhai, Y.; Gu, H. Dynamic modeling of quadrotor AUV using a novel CFD simulation. *Ocean. Eng.* **2021**, *237*, 109651. [CrossRef]
20. Moshayedi, A.J.; Gheibollahi, M.; Liao, L. The quadrotor dynamic modeling and study of meta-heuristic algorithms performance on optimization of PID controller index to control angles and tracking the route. *IAES Int. J. Robot. Autom.* **2020**, *9*, 256. [CrossRef]
21. Almakhles, D.J. Robust backstepping sliding mode control for a quadrotor trajectory tracking application. *IEEE Access* **2019**, *8*, 5515–5525. [CrossRef]
22. Lambert, N.O.; Drew, D.S.; Yaconelli, J.; Levine, S.; Calandra, R.; Pister, K.S. Low-level control of a quadrotor with deep model-based reinforcement learning. *IEEE Robot. Autom. Lett.* **2019**, *4*, 4224–4230. [CrossRef]
23. Yang, S.; Xian, B. Energy-based nonlinear adaptive control design for the quadrotor UAV system with a suspended payload. *IEEE Trans. Ind. Electron.* **2019**, *67*, 2054–2064. [CrossRef]
24. Zhang, X.; Wang, Y.; Zhu, G.; Chen, X.; Li, Z.; Wang, C.; Su, C.Y. Compound adaptive fuzzy quantized control for quadrotor and its experimental verification. *IEEE Trans. Cybern.* **2020**, *51*, 1121–1133. [CrossRef] [PubMed]
25. Wang, L. Review of the application of open source flight control in multi-rotor aircraft. *Int. Core J. Eng.* **2021**, *7*, 261–270.
26. García, J.; Molina, J.M. Simulation in real conditions of navigation and obstacle avoidance with PX4/Gazebo platform. *Pers. Ubiquitous Comput.* **2020**, *26*, 1171–1191. [CrossRef]
27. Kawamura, E.; Azimov, D. Integrated extremal control and explicit guidance for quadcopters. *J. Intell. Robot. Syst.* **2020**, *100*, 1583–1613. [CrossRef]
28. Khosiawan, Y.; Nielsen, I. A system of UAV application in indoor environment. *Prod. Manuf. Res.* **2016**, *4*, 2–22. [CrossRef]
29. Zhang, T.; Liu, C.; Li, J.; Pang, M.; Wang, M. A new visual inertial Simultaneous Localization and Mapping (SLAM) algorithm based on point and line features. *Drones* **2022**, *6*, 23. [CrossRef]
30. Bauersfeld, L.; Ducard, G. RTOB SLAM: Real-time onboard laser-based localization and mapping. *Vehicles* **2021**, *3*, 778–789. [CrossRef]

31. Jing, Y.; Qi, F.; Yang, F.; Cao, Y.; Zhu, M.; Li, Z.; Lei, T.; Xia, J.; Wang, J.; Lu, G. Respiration detection of ground injured human target using UWB radar mounted on a hovering UAV. *Drones* **2022**, *6*, 235. [CrossRef]
32. Kwon, J.; Hailes, S. In Scheduling UAVs to bridge communications in delay-tolerant networks using real-time scheduling analysis techniques. In Proceedings of the 2014 IEEE/SICE International Symposium on System Integration, Tokyo, Japan, 13–15 December 2014; IEEE: Piscataway, NJ, USA, 2014; pp. 363–369.
33. Potortì, F.; Palumbo, F.; Crivello, A. Sensors and sensing technologies for indoor positioning and indoor navigation. *Sensors* **2020**, *20*, 5924. [CrossRef]
34. Steenbeek, A.; Nex, F. CNN-based dense monocular visual SLAM for real-time UAV exploration in emergency conditions. *Drones* **2022**, *6*, 79. [CrossRef]
35. Moussa, M.; Zahran, S.; Mostafa, M.; Moussa, A.; El-Sheimy, N.; Elhabiby, M. Optical and mass flow sensors for aiding vehicle navigation in gnss denied environment. *Sensors* **2020**, *20*, 6567. [CrossRef] [PubMed]
36. Deng, H.; Fu, Q.; Quan, Q.; Yang, K.; Cai, K.-Y. Indoor multi-camera-based testbed for 3-D tracking and control of UAVs. *IEEE Trans. Instrum. Meas.* **2019**, *69*, 3139–3156. [CrossRef]
37. Xin, C.; Wu, G.; Zhang, C.; Chen, K.; Wang, J.; Wang, X. Research on indoor navigation system of uav based on lidar. In Proceedings of the 2020 12th International Conference on Measuring Technology and Mechatronics Automation (ICMTMA), Phuket, Thailand, 28–29 February 2020; IEEE: Piscataway, NJ, USA, 2020; pp. 763–766.
38. Youn, W.; Ko, H.; Choi, H.; Choi, I.; Baek, J.-H.; Myung, H. Collision-free autonomous navigation of a small UAV using low-cost sensors in GPS-denied environments. *Int. J. Control. Autom. Syst.* **2021**, *19*, 953–968. [CrossRef]
39. Majumdar, A.; Ahmadi, A.A.; Tedrake, R. Control design along trajectories with sums of squares programming. In Proceedings of the 2013 IEEE International Conference on Robotics and Automation, Karlsruhe, Germany, 6–10 May 2013; IEEE: Piscataway, NJ, USA, 2013; pp. 4054–4061.
40. Meyer-Delius, D.; Beinhofer, M.; Burgard, W. Occupancy grid models for robot mapping in changing environments. In Proceedings of the 26th AAAI Conference on Artificial Intelligence, Toronto, ON, Canada, 22–26 July 2012.
41. Dhulkefl, E.; Durdu, A.; Terzioğlu, H. Dijkstra algorithm using uav path planning. *Konya Mühendislik Bilimleri Dergisi* **2020**, *8*, 92–105.
42. Zhou, F.; Nie, H. Quick path planning based on shortest path algorithm for multi-uav system in windy condition. *Control. Syst. Eng.* **2021**, preprint.
43. Hong, Z.; Sun, P.; Tong, X.; Pan, H.; Zhou, R.; Zhang, Y.; Han, Y.; Wang, J.; Yang, S.; Xu, L. Improved a-star algorithm for long-distance off-road path planning using terrain data map. *ISPRS Int. J. Geo-Inf.* **2021**, *10*, 785. [CrossRef]
44. Belanová, D.; Mach, M.; Sinčák, P.; Yoshida, K. Path planning on robot based on D* lite algorithm. In *Procedings of the 2018 World Symposium on Digital Intelligence for Systems and Machines (DISA), Košice, Slovakia, 23–25 August 2018*; IEEE: Piscataway, NJ, USA, 2018; pp. 125–130.
45. Chen, X.; Zhang, J. The three-dimension path planning of UAV based on improved artificial potential field in dynamic environment. In Proceedings of the 2013 5th International Conference on Intelligent Human-Machine Systems and Cybernetics, Hangzhou, China, 26–27 August 2013; IEEE: Piscataway, NJ, USA, 2013; pp. 144–147.
46. Li, F.; Zlatanova, S.; Koopman, M.; Bai, X.; Diakité, A. Universal path planning for an indoor drone. *Autom. Constr.* **2018**, *95*, 275–283. [CrossRef]
47. González de Santos, L.M.; Frías Nores, E.; Martínez Sánchez, J.; González Jorge, H. Indoor path-planning algorithm for uav-based contact inspection. *Sensors* **2021**, *21*, 642. [CrossRef]
48. Xu, Z.; Zhang, L.; Ma, X.; Liu, Y.; Yang, L.; Yang, F. An anti-disturbance resilience enhanced algorithm for UAV 3D route planning. *Sensors* **2022**, *22*, 2151. [CrossRef]
49. Hao, K.; Zhao, J.; Yu, K.; Li, C.; Wang, C. Path planning of mobile robots based on a multi-population migration genetic algorithm. *Sensors* **2020**, *20*, 5873. [CrossRef]
50. Ajeil, F.H.; Ibraheem, I.K.; Azar, A.T.; Humaidi, A.J. Grid-based mobile robot path planning using aging-based ant colony optimization algorithm in static and dynamic environments. *Sensors* **2020**, *20*, 1880. [CrossRef] [PubMed]
51. Li, J.; Sun, T.; Huang, X.; Ma, L.; Lin, Q.; Chen, J.; Leung, V.C. A memetic path planning algorithm for unmanned air/ground vehicle cooperative detection systems. *IEEE Trans. Autom. Sci. Eng.* **2021**, *19*, 2724–2737. [CrossRef]
52. Liu, G.; Shu, C.; Liang, Z.; Peng, B.; Cheng, L. A modified sparrow search algorithm with application in 3d route planning for UAV. *Sensors* **2021**, *21*, 1224. [CrossRef]
53. Wang, X.; Pan, J.S.; Yang, Q.; Kong, L.; Snášel, V.; Chu, S.C. Modified mayfly algorithm for UAV path planning. *Drones* **2022**, *6*, 134. [CrossRef]
54. Li, S.; Zhang, H.; Li, Z.; Liu, H. An air route network planning model of logistics UAV terminal distribution in urban low altitude airspace. *Sustainability* **2021**, *13*, 13079. [CrossRef]
55. Yin, G.; Zhou, S.; Wu, Q. An improved RRT algorithm for UAV path planning. *Acta Electonica Sin.* **2017**, *45*, 1764.
56. Pettersson, P.O.; Doherty, P. Probabilistic roadmap based path planning for an autonomous unmanned aerial vehicle. In Proceedings of the ICAPS-04 Workshop on Connecting Planning Theory with Practice, Whistler, Canada, 3–7 June 2004.
57. Gang, L.; Wang, J. PRM path planning optimization algorithm research. *Wseas Trans. Syst. Control.* **2016**, *11*, 81–86.

58. Chen, J.; Zhou, Y.; Gong, J.; Deng, Y. An improved probabilistic roadmap algorithm with potential field function for path planning of quadrotor. In Proceedings of the 2019 Chinese Control Conference (CCC), Guangzhou, China, 27–30 July 2019; IEEE: Piscataway, NJ, USA, 2019; pp. 3248–3253.
59. Mohanta, J.C.; Keshari, A. A knowledge based fuzzy-probabilistic roadmap method for mobile robot navigation. *Appl. Soft Comput.* **2019**, *79*, 391–409. [CrossRef]
60. Raheem, F.A.; Abdulakareem, M.I. Development of a* algorithm for robot path planning based on modified probabilistic roadmap and artificial potential field. *J. Eng. Sci. Technol* **2020**, *15*, 3034–3054.
61. Abdulakareem, M.I.; Raheem, F.A. Development of path planning algorithm using probabilistic roadmap based on ant colony optimization. *Eng. Technol. J.* **2020**, *38*, 343–351. [CrossRef]
62. Che, E.; Jung, J.; Olsen, M.J. Object recognition, segmentation, and classification of mobile laser scanning point clouds: A state of the art review. *Sensors* **2019**, *19*, 810. [CrossRef] [PubMed]
63. Rahul Kala's Source Codes. Available online: http://rkala.in/codes.php (accessed on 5 May 2022).
64. Point Cloud Library. Available online: https://pointclouds.org/ (accessed on 20 April 2022).