drones

MDPI

Article

Vision Object-Oriented Augmented Sampling-Based Autonomous Navigation for Micro Aerial Vehicles

Xishuang Zhao [1], Jingzheng Chong [1], Xiaohan Qi [1] and Zhihua Yang [1,2,*]

[1] Communication Engineering Research Center, Harbin Institute of Technology, Shenzhen 518000, China; 19s152074@stu.hit.edu.cn (X.Z.); chongjz@stu.hit.edu.cn (J.C.); qixiaohan@stu.hit.edu.cn (X.Q.)
[2] Pengcheng Lab, Shenzhen 518000, China
* Correspondence: yangzhihua@hit.edu.cn

Abstract: Autonomous navigation of micro aerial vehicles in unknown environments not only requires exploring their time-varying surroundings, but also ensuring the complete safety of flights at all times. The current research addresses estimation of the potential exploration value neglect of safety issues, especially in situations with a cluttered environment and no prior knowledge. To address this issue, we propose a vision object-oriented autonomous navigation method for environment exploration, which develops a B-spline function-based local trajectory re-planning algorithm by extracting spatial-structure information and selecting temporary target points. The proposed method is evaluated in a variety of cluttered environments, such as forests, building areas, and mines. The experimental results show that the proposed autonomous navigation system can effectively complete the global trajectory, during which an appropriate safe distance could always be maintained from multiple obstacles in the environment.

Keywords: vision-based navigation; path planning; sampling; B-spline

Citation: Zhao, X.; Chong, J.; Qi, X.; Yang, Z. Vision Object-Oriented Augmented Sampling-Based Autonomous Navigation for Micro Aerial Vehicles. *Drones* **2021**, *5*, 107. https://doi.org/10.3390/drones5040107

Academic Editor: George Nikolakopoulos

Received: 14 August 2021
Accepted: 23 September 2021
Published: 30 September 2021

1. Introduction

Due to their good flexibility, strong maneuverability, easy operation, and unique viewpoint, micro aerial vehicles (MAVs) are widely used in aerial photography, search-and-rescue missions [1], delivery of goods, and mine exploration [2], etc. The autonomous navigation systems of drones can replace humans to complete some tasks that are dangerous or impractical for humans. The research on autonomous navigation systems can effectively reduce the complexity of tasks and improve the efficiency of their execution. It has attracted widespread attention in recent years. For example, in search-and-rescue missions, an autonomous navigation system can autonomously control the MAV to avoid obstacles in real time and provide image information with a unique perspective for rescue operations.

Typically, an autonomous navigation system consists of four parts: perception, mapping, planning and control, in which the planning provides a real-time, safe, collision-free trajectory for MAVs that meets dynamic constraints. At the present, the research of planning based on target searches in unknown environments mainly deals with two problems: the first is obstacle avoidance, while the second is solving the problem of getting stuck at local minima. This poses a special problem in unexplored or partially unexplored environments, where only locally optimal or reactive planners will frequently fail to find a path [3]. Different from the known environment, an autonomous navigation system in partially known or unknown environments will model the environment through sensors step by step. Since the environmental information is gradually obtained, there is a high risk of possible collision in the trajectory at the current and next moments. Therefore, the planning policy should not only gradually complete the task through exploration, but also have the capability of guaranteeing the safety of the MAV in unknown surroundings.

However, current algorithms cannot fully meet this requirement. The proposed scheme simplifies the environment into a model with sparse and evenly distributed obstacles and makes optimistic assumptions about it [4]. This method cannot make immediate adjustments in the presence of large, irregular obstacles, although it is indispensable in a real environment. The author proposes that the trajectory follows obstacles and always maintains a safety range [5] when MAVs face complex environments. Although this method guarantees sufficient safety, planning is restricted. It is not only unfavorable to the execution of the task, but also makes planning get stuck at local minima, and can even fail to find a path. Another type of approach does not focus on obstacles, but ensures safety from the perspective of local re-planning. Some scholars combine the strategy of exploration and re-planning, which has shown good results. However, the global trajectory is mechanically spliced between the previous and next trajectories, and so lacks flexibility. This leads to discontinuities in the trajectory direction, which greatly increases the possibility of drone collisions.

Based on the rich information from the vision sensor, in this work we propose a model for extracting spatial-structural information, which can guide drones well to maintain a suitable safe distance with obstacles in space. With it, we develop an exploration strategy with a B-spline-based local trajectory optimization by selecting intermediate goal points to ensure a sufficiently smooth global trajectory. Additionally, we quantitatively compare our autonomous navigation system with the Ewok system [6] by simulating a variety of different environments for the system to operate in, including three specific scenarios: a dense forest, building and mine. Experiments show that our planning algorithm can not only achieve a better connection between the two trajectories, but also maintain a high success rate in a variety of complex environments.

2. Related Work

At the present, there are many studies on path planning strategies in unknown environments, which can be divided into three categories: motion primitive-based approaches,trajectory optimization approaches and sampling-based approaches.

Motion primitive-based approaches discretize the state space of the drone into a variety of motion primitives and then search for the corresponding operation in the motion primitive library according to the current state of the drone. The approach makes an immediate decision based on the sensor data without building a persistent map [7–9]. This method has the advantages of a fast response and a good effect on moving objects, but it can only run in a scenario with low obstacle density, and it easy for it to get stuck in the local optimal position in a complex obstacle environment.

Trajectory optimization approaches rely on a minimized objective function to achieve a collision-free and smooth trajectory through multiple iterations of optimization. CHOP [10] and CTTOP [11] both use a two-part objective function with a smoothness and collision cost. The difference is that the former uses a series of discrete points to express the trajectory and the latter uses polynomial segments instead. A more recent advent in trajectory optimization expresses the trajectory by B-spline and performs gradient descent with positions of discrete waypoints as parameters [6]. A trajectory optimization-based planner requires an initial trajectory. If the initial trajectory passes through a large obstacle, the ability of this optimizer will be very limited, and the trajectory cannot be guaranteed to be collision-free. Therefore, it is not suitable for large-scale path planning.

The closest to our proposed approach are search-based approaches. Traditional methods, such as RRT [12] and RRBT [13], obtain samples from graphs by random sampling, collision detection, expansion of the RRT tree, and repeated iterations. Finally, a non-smooth path from the start point to the end point is constructed. This method initially needs to search in the global map, so an a priori map is necessary. In order to adapt to the unknown environment, scholars have proposed a strategy of exploration, which does not focus on collision avoidance, but tries to discover the unexplored space. The proposed receding horizon Next-Best-View Planner (NBVP) was proposed by Bircher in [14], which

counted the number of unknown occupied cells in the field of view of the sensor as the potential exploration value of this sample point and established the RRT tree. Each time, the MAV only executed the branch that could obtain the highest profit. Anna [15] uses the storage characteristics of an Octomap to extract the frontier instead of the complex calculation of the traditional method and incorporates the idea of exploration gain based on the sampling method, which achieved a good effect in the exploration map. Another work saves the explored historical information of NBVP so as to achieve a high sampling efficiency [16]. In an unknown environment, due to the limited perception range of the sensor, the contribution of RRT to planning is limited, requiring a lot of calculations to maintain the RRT tree. Oleynikova et al. [3] extend the method of NBVP and replace the method of RRT tree with the strategy of intermediate goal, by proposing a local trajectory optimization method represented trajectory based on a polynomial. Although the exploration strategy of intermediate goal is very effective in guiding drones to explore in unknown environments, the local plan optimizer expressing the trajectory in a polynomial obtained the global trajectory with insufficient smoothness and many inflection points, especially in the connection point between two segments trajectories. The proposed algorithm in this paper extends the exploration strategy of intermediate goal, but replaces random sampling by extracting samples from the special spatial-structure information; in particular, the proposed algorithm expresses the trajectory based on B-spline that has better locality for connecting two segments trajectories and achieves sufficient smoothness of the global trajectory.

3. System Framework

In order to make the autonomous navigation system adapt to different application scenarios, we assume a task scenario in which a drone needs to explore from the source point (x_s, y_s, z_s) to the goal point (x_g, y_g, z_g). There is a space $V \in R^3$ without prior information, where the obstacles are densely distributed and unevenly distributed. The drone is equipped with an autonomous navigation system and vision sensor. The vision sensor can obtain an RGB image and depth image in the field of view of the drone at the current position, but the effective range of depth information is limited, as shown in Figure 1.

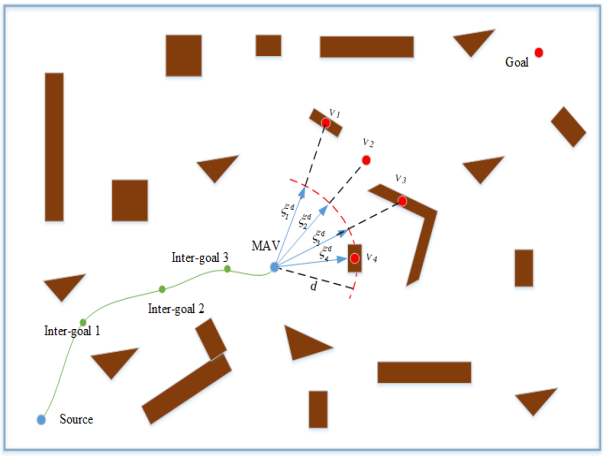


Figure 1. MAV mission execution scenario. Source point (blue) is the drone take-off position, while goal point (red) is the goal position. Obstacles (brown) of various shapes are unevenly distributed in the space. The effective depth distance of the vision sensor can only reach d. In particular, $v1$, $v2$, $v3$, and $v4$ (red point) are the viewpoints extracted from the image, which are filtered to obtain the *Inter–goal* point (green). Under the guidance of *Inter–goal*, the local planning optimizer obtains the trajectory (green line).

The autonomous navigation system is mainly composed of four parts, perception, mapping, planning and control; in this work, we focus on the part of planning. Since there is no prior environmental information, MAVs need to continuously model environmental information through vision sensors, called perception. Vision-based autonomous navigation systems generally obtain visual information from a vision camera that can be an RGB-D camera, monocular or stereo. Visual simultaneous localization and mapping (V-SLAM) technology is often used to provide the drone's pose and map, which is mainly divided into two parts: localization and mapping. Since our research is mainly focused on the planning part, we provide drone poses through high-precision IMU and obtain the grid map through ray-cast operations [6,17], which is a typical mapping algorithm. At the same time, we propose a special spatial-structure information extraction model, which will input the two-dimensional image and output a special spatial-structure information ξ. Planning plans a collision-free path and satisfies a series of constraints trajectory for MAV. In this part, we have improved the sampling method to enhance the applicability of the autonomous navigation system in the actual environment. The intermediate goal selected by our planning algorithm is more instructive and safer. Local planning optimizer can adjust flexibly in real-time, when the drone faces unmodeled, large and irregular obstacles. Control receives control information from planning to guide drone flight, as shown in Figure 2.

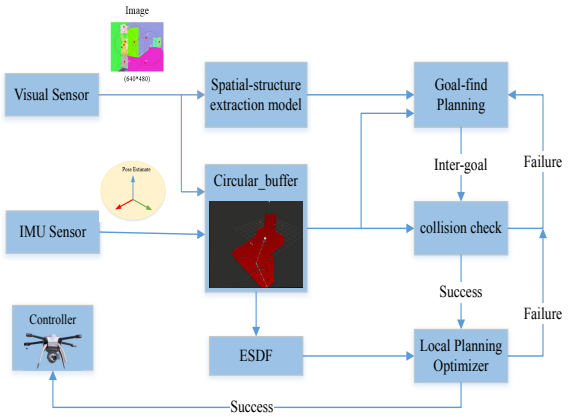


Figure 2. Autonomous navigation system. The spatial-structure extraction model obtains image data from the visual-sensor, and sends it to goal-find planning. Goal-find planning quantifies the exploration and safety values of each viewpoint, selects the best inter-goal and detects collision. Then, local planning optimizer relies on inter-goal guidance to plan the trajectory.

4. Methods

Our path planning method is an extension of intermediate goal strategy [3], which belongs to the sampling method. Intermediate goal strategy compares a large number of randomly sampled points to obtain a position with a greater exploration value. However, the sample points obtained by random sampling contain limited spatial information, which cannot well express the relationship between the drone and the objects in space. It is difficult for such an intermediate goal point to ensure the safety of the drone in an unknown environment. To deal with these problems, we propose a spatial-structure extraction model, which will process the two-dimensional image and output special spatial-structure information. The special information can be expressed as a ray or a viewpoint in R^3, which are limited in number and contain information about the distribution of objects in space. Then, we improve the original value evaluation function by adding a safety evaluation factor, balance the value of exploration and safety with a weight factor, and adjust the number of candidate intermediate goal points in an environment-adaptive manner. Last but not least, we propose setting the intermediate goal for the initialization parameter of the local planning optimizer, which expresses the trajectory in B-spline. In order to facilitate the evaluation of the value of exploration and safety, we use the circular

buffer, a kind of occupancy grid map, to store the environmental information obtained by the sensor, as shown in Figure 3.

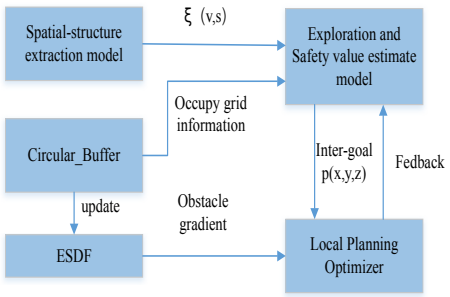


Figure 3. The relationship between the modules. The exploration-safety value estimation model obtains the sample point from the spatial-structure extraction model and queries the surrounding environment information in the circular buffer. The local planning optimizer takes *inter–goal* as intermediate goal point and optimizes trajectory with collision costs under the support of ESDF.

4.1. Map Representation

Building a grid map is a very common strategy used to facilitate collision detection in planning. Environmental information is projected onto the grid map through the sensor. Objects in space are represented in the grid in an occupied state. One grid also includes occupied and unknown states. Usenko compares a kind of 3D circular buffer [6] and the Octomap [18]. The three-dimensional Circular Buffer is a kind of grid map that defines a variable $offset$ and a continuous $array$ with a size of 2 N. The $offset$ is given by the drone position, and $array$ stores the state of each voxel in the corresponding space with the $offset$ as the center. The circular buffer maintains a higher rate and higher quality for point cloud insertion and occupancy information query, which can speed up the estimation of exploration value and safety value. So, we set the three-dimensional circular buffer as a storage method for local environment information.

$$Array = \{(n_x, n_y, n_z) | -N < n_x, n_y, n_z < N\} \tag{1}$$

$$\begin{bmatrix} n_x \\ n_y \\ n_z \end{bmatrix} = \left[\frac{1}{Res} \left(\begin{bmatrix} x \\ y \\ z \end{bmatrix} - \begin{bmatrix} o_x \\ o_y \\ o_z \end{bmatrix} \right) \right]_{Round} \tag{2}$$

where $Offset(o_x, o_y, o_z)$ is constantly updated by the drone position. In particular, (n_x, n_y, n_z) is the index in the $Array$ corresponding to the spatial coordinate (x, y, z).

We proposed a spatial-structure information extraction model, in which this special information expresses a direction relationship between the position of the MAV and the geometric center of the object or hole in MAV's field of view. We define ξ as the special structure information, which represents multiple rays from the position of the drone to its viewpoints. The direction of the ray is the geometric center of one object or one hole formed by multiple objects.

$$\xi(i) = \left\{ (view(i), s) | view(i), s \in R^3, i = 1, 2, \ldots, n \right\} \tag{3}$$

4.2. Spatial-Structure Information Extraction

The ray-cast method [6] can efficiently insert the point cloud data of the visual sensor into the circular buffer and update the occupancy probability of each voxel. However, the detection distance range of the vision sensor is limited, so only the point cloud information within the detection range is beneficial to the circular buffer. We design a spatial-structure information extraction model based on image segmentation technology to extract environmental information. The method can extract information outside of the valid depth range of the visual sensor from the image. It is not a traditional one-to-one mapping of voxels in space and does not require the support of depth information of vision sensor. Instead, it uses the overall characteristic of a single object in space presented in the image data to map the spatial structure of the object from the outline.

The spatial-structure extraction algorithm is divided into three steps: cutting image, extracting center-point, and converting reference system. Cutting image is the core of the spatial-structure information extraction algorithm. We define a single image Fig, and a single $Block$, which represents a collection of pixels with the same characteristic in the image. We segmented the image based on the graph-based image segmentation algorithm [19], which abstracts the pixels in the image into vertices and edges in the graph theory, and divides the graph into multiple regions C by the clustering method. The algorithm gives three parameters, $Sigma$, k, and Min to, respectively, control the size, color sensitivity and minimum block of the segmented image.

$$Fig = \{p_{u,v} | 0 \leq u < row, 0 \leq v < column\} \tag{4}$$

$$Block = \{p_{u,v} | p_{u,v} \in Fig \cap p_{u,v} \in C\} \tag{5}$$

After cutting, $Block$ corresponds to all the pixels contained in the regional C in the graph, which incorporates different objects or gaps formed by the extrusion of multiple objects. In the extracting center-point part, we calculated the smallest circumscribed circle of each $Block$ and extracted the center points (u, v). Since (u, v) is only extracted from the RGB image, the depth information is lost. In order to facilitate the calculation, we fixed $z_c = C$ as a constant, which is more appropriate to use the effective detection range of the vision sensor. One 3D viewpoint $view(x, y, z)$ will be obtained in the drone world reference by converting the reference system. The conversion is shown in Equations (6) and (7).

$$\left\{ \begin{array}{l} x_c = \frac{(u-cx)*z}{fx} \\ y_c = \frac{(v-cy)*z}{fy} \end{array} \right| _{z_c=C} \tag{6}$$

$$[x, y, z, 1]^\intercal = T \cdot [x_c, y_c, z_c, 1]^\intercal \tag{7}$$

where cx, cy, fx, and fy are four camera internal parameters and T is the transform matrix.

From above, we obtain some viewpoints $view(x, y, z = c)$ with spatial-structure information from the drone's field of view. It is used to represent a ray ξ, which starts from Dp and points to the $view$ that is the geometrical center of an object or an gap, as shown in Equation (8). The spatial-structure information extraction is shown in Algorithm 1.

$$\xi(i) = \Big\{ (view(i), Dp) | view(i), Dp \in R^3, i = 1, 2, \ldots, n \Big\}. \tag{8}$$

where Dp is the drone's position, and $view$ is the viewpoint obtained from cutting fig. The number of viewpoints is n.

Algorithm 1 Extracting Spatial-structure Information.

Input: $F(f_{i,j}) = fig, s = (sigma, k, min), k = (cx, cy, fx, fy)$
Output: *view*

$S\{vi|i = 1, 2, \ldots, M\} \leftarrow graphBaseSegmetation(F, sigma, k, min)$
$T \leftarrow getDroneTransformMatrix$
for $i = 1 : 1 : M$ **do**
  $center[u, v] \leftarrow mean(ci)$
  $x_c = \frac{(u-cx)*z}{fx}|_{z=C}$
  $y_c = \frac{(v-cy)*z}{fy}|_{z=C}$
  $v_c = (x_c, y_c, z_c)$
  $v \leftarrow converReferenceWorld(T, v_c)$
  $view(i) \leftarrow add(v)$
end for

4.3. Intermediate Goal

We combined the special spatial-structure information ξ to improve the intermediate goal strategy [3]. Since the special spatial-structure information always points to the geometric center of the object or the gap, it can find an optimal intermediate goal point very effectively and control the number of candidate seeds through the parameters of the spatial-structure information extraction model.

The planning algorithm [3,14,20] sets the potential exploration value as a cost function in order to escape from the local optimal solution when selecting the next intermediate goal points. After analysis, the method that simply sets the exploration value as an indicator can easily lead drones into dangerous scenarios. Combining the proposed spatial-structure information ξ, we divide the cost function Dv into two parts. One part is the exploration obtained by counting the number of grids in an unknown state in the circular buffer map. The other part is safety obtained by counting the number of grids in an unoccupied state in the circular buffer map. We balance the two indicators of exploration and safety by different weight parameters.

$$ex(Dp, \xi) = \{v|v \in frustum(Dp, \xi) \cap v \in unknow(v)\} \tag{9}$$

$$sa(Dp, \xi) = \{v|v \in frustum(Dp, \xi) \cap v \in free(v)\} \tag{10}$$

$$D_v = \lambda_s sa(Dp, \xi) + \lambda_e ex(Dp, \xi) \tag{11}$$

where λ_s and λ_e are the weighting parameter to balance the two values of exploration and safety in the cost function Dv. In particular, Dp is the drone's position, while ξ represents the special spatial-structure information.

Vision sensors have a fixed view field FOV, which limits the observation range of a single frame of image in the horizontal plane. The strategy of intermediate goal just obtains the candidate seeds by the camera's view field at a certain range. The adaptive control of the number of viewpoints expands the drone's view field. The method is used to adaptively control the number of viewpoints through the environment and store them in the inter-goal buffer. The intermediate goal algorithm is shown in Algorithm 2.

We set a threshold for the cost function Dv. Viewpoints that exceeded the threshold were all saved in the inter-goal buffer, while the others were discarded. This threshold is dynamically updated in each iteration. The advantage of self-adaptation is that the number of viewpoints can be adjusted in real time according to the environmental conditions, and the Dv value of viewpoint can be kept stable.

$$NextTHR = \alpha \cdot THR + \frac{\beta}{2} \cdot (DvBuffer[num - 1] + DvBuffer[num]) \tag{12}$$

$$\alpha + \beta = 1 \tag{13}$$

where $DvBuffer$ is the set of Dv values for each viewpoint, and num is the number of viewpoints expected to be retained. α and β are two respective weight parameters. The larger the α, the more stable the number of viewpoints and the less flexibility.

Algorithm 2 Obtain Intermediate Goal.

Input: $goal = (x_g, y_g, z_g), Dp = (x_d, y_d, z_d), N, S = \{vi | i = 1, 2, \ldots, M\}$
Output: $InterGoalBuffer$
for $i = 1 : 1 : M$ **do**
  $[yaw, pich] \leftarrow yawPichExtracte(Dp, Vi, Goal)$
  $Dvalue = \alpha \cdot yaw + \beta \cdot pich$
  $Vpoint(i) \leftarrow add((vi, Dvalue))$
end for
$Vpoint \leftarrow sort(Vpoint)$
$\xi = (Dp, Vpoint)$
$N = max(length(\xi), N)$
for $i = 1 : 1 : N$ **do**
  $[ex, sa] \leftarrow countNumberOfGrid(\xi(i))$
  $Lvalue = wEx * ex + wSa * sa$
  $InterGoalBuffer(i) \leftarrow add((\xi(i), Lvalue))$
end for
$InterGoalBuffer \leftarrow sort(InterGoalBuffer)$

4.4. Flight Status Switch

We obtained the intermediate goal ξ by cost function, which contains a specific direction from the drone position in R^3. In order to provide a three-dimensional point for local planning to optimize the local trajectory, we added a radius r to ξ and set three types of radius lengths to divide the flight state into three types. The longer the radius r, the smaller the number of intermediate goals. However it is difficult to find a feasible solution in the environment with dense obstacles. As a result, it is easy for planning to fall into a local optimal solution. On the other side, the shorter the radius r, the more intermediate goals, and the more time it takes to iterate an intermediate goal. However, in the gradual short-distance exploration, more feasible solutions can be searched to ensure that the MAV reaches the final goal.

The first flight state is called the Forward state, which is designed to pass through the scene with sparse obstacles. The flight step length r is set to be as long as possible to avoid the complexity caused by frequent switching of intermediate goal. The second state is called the exploratory state, designed to pass through dense obstacles, in which the planning algorithm will frequently fail to find a path. The step length r is set to be shorter in order to explore the environment and eliminate visual blind spots step by step. The third state is designed to control the direction of the MAV's view field. The flight state switch algorithm is shown in Algorithm 3.

In order to avoid detours and ensure the MAV advances to the goal, we set the state to keep the MAV's yaw angle as far as possible to the goal direction and propose a simplified method to estimate the deviation between the intermediate goal and the goal in the yaw angle and pitch angle directions, as shown in Figure 4. The reward function Lv is used and defined as follows.

$$Lv = wYaw \cdot y_{vsg} + wPich \cdot p_{vsg} \tag{14}$$

$$y_{vsg} = acos \frac{(x_i - x_s, y_i - y_s)(x_g - x_s, y_g - y_s)}{\sqrt{(x_i - x_s)^2 + (y_i - y_s)^2}\sqrt{(x_g - x_s)^2 + (y_g - y_s)^2}} \tag{15}$$

$$y_{vsg} = acos \frac{(y_i - y_s, z_i - z_s)(y_g - y_s, z_g - z_s)}{\sqrt{(y_i - y_s)^2 + (z_i - z_s)^2}\sqrt{(y_g - y_s)^2 + (z_g - z_s)^2}} \tag{16}$$

where y_{vsg} is the angle formed by the intermediate goal point $\xi_r = (x_\xi, y_\xi, z_\xi)$, the drone position point $Dp(x_s, y_s, z_s)$, and the goal point $goal(x_g, y_g, z_g)$ on the horizontal plane. p_{vsg} is the angle formed by ξ_r, Dp and $goal$ on the vertical plane. $wYaw$ and $wPich$ are weights parameters.

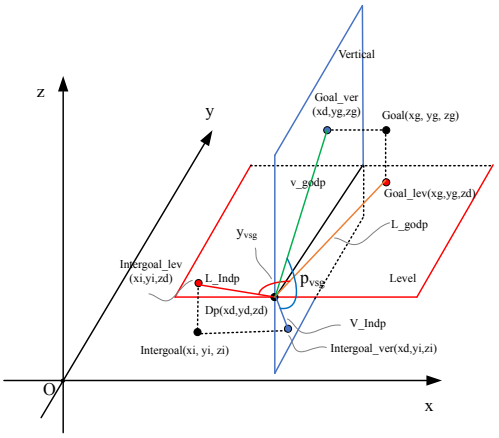


Figure 4. Pitch and yaw angle deviation model. Dp point is the drone position, $Intergoal$ point is the intermediate goal, and $Goal$ point is the goal. $yvsg$ is the angle between line L_{Indp} (red line) and L_{godp} (brown line), while p_{vsg} is the angle between line V_{Indp} (blue line) and V_{godp} (green line).

4.5. Local Planning Optimization

The local planning optimization algorithm takes the global path as the initial trajectory input [20,21] or initializes a straight line from the starting point to the goal point as the input [3,6,11,14]. Then, it penalizes the deviation from the initial trajectory in the cost function to ensure that the local path planning is around the initial trajectory. Although the method greatly ensures that the local trajectory follows the initial trajectory, it also limits the optimization range of the local trajectory, lacks flexibility, and cannot effectively deal with the emergency goal point change. We use b-spline to express the trajectory, other than the traditional mode of taking initialization trajectory as input. The intermediate goal point is directly used as a part of the objective function of the local planning. We divided the penalty function of trajectory optimization into three parts: the first part is the intermediate goal cost function that penalizes the trajectory deviating from the intermediate goal, the second part is the collision cost, and the third part is the trajectory smooth cost function.

$$Etotal = Eg + Ec + Es \tag{17}$$

$$Eg = \lambda_g (p(t_{ep}) - \xi_r)^2 \tag{18}$$

$$Ec = \lambda_c \int_{tmin}^{tmax} sg(d(p(t)) - \tau)(d(p(t)) - \tau)^2 ||p'(t)|| d_t \tag{19}$$

$$Es = \lambda_s \sum_{i=2}^{4} \int_{tmin}^{tmax} (p^{(i)}(t))^2 + sg(p^{(i)}(t) - pmax) e^{(p^{(i)}(t))^2 - (p^i_{max})^2 - 1} d_t \tag{20}$$

$$sg = \frac{1}{2}(sgn(x) + 1) \tag{21}$$

where $p(t)$ is the drone position at time t. ξ_r is the intermediate goal point intercepted from the ray of radius r. $d(p(t))$ is the distance function from the drone position to the obstacle, which is directly obtained by ESDF. Es is the penalty integral over square derivatives of the trajectory to smooth the trajectory. It set an exponential function to limit the maximum speed and maximum acceleration, which was proposed by Usenko in [6]. $sgn(x)$ is a symbolic function. λ_g, λ_c and λ_s are three weight parameters.

Algorithm 3 Status Swich.

Input: $goal = (xg, yg, zg), d = (d_f, d_e, d_c), \varepsilon$
Output: $State$
$State \Leftarrow Forward$
$Dp \leftarrow updateDonePostion$
while $Dp \neq goal$ **do**
$InterGoal \Leftarrow InterGoalBuffer[0]$
switch ($State$)
case $Forward$**:**
$p \leftarrow cutPointInLine(InterGoal, d_f)$
$path(t) \leftarrow LocalPlanning(Dp, goal, p)$
if $CheckPathBlock(path) = \phi$ **then**
$State \Leftarrow Exploration$
$InterGoalBuffer \leftarrow Delet(InterGoal)$
end if
if $CheckViewRange(path) > \varepsilon$ **then**
$State \Leftarrow ForceContral$
end if
case $Exploration$**:**
$p = cutPointInLine(InterGoal, d_e)$
$path(t) \leftarrow LocalPlanning(p)$
if $CheckPathBlock(path) \neq \phi$ **then**
$State \Leftarrow ForceContral$
end if
case $ForceContral$**:**
$p = cutPointInLine(InterGoal, d_c)$
$path(t) \leftarrow LocalPlanning(p)$
if $CheckViewRange(path) < \varepsilon$ **then**
$State \Leftarrow Forward$
end if
if $CheckPathBlock(path) \neq \phi$ **then**
$State \Leftarrow Exploration$
end if
end switch
end while

The papers [22,23] found that B-spline has sufficient local characteristics. When the local control point changes, the new trajectory can be expressed as $q(t)$.

$$\begin{aligned} q(t) &= \sum_{i=0}^{j-1} p_i N_{i,k}(t) + (v + p_i) N_{i,k}(t) + \sum_{i=j+1}^{n} p_i N_{i,k}(t) \\ &= p(t) + v N_{i,k}(t) \end{aligned} \tag{22}$$

The new curve $q(t)$ is the sum of the original curve $p(t)$ and a translation vector $vN_{i,k}(t)$,which is non-zero on the interval $[t_i, t_{i+k+1})$. Hence, moving a control point only affects the shape of a section of the given curve. It can be seen from Equation (22) that when the intermediate goal points are alternated or urgently replaced, only the optimized local trajectory in the trajectory will change, and the optimized local trajectory will be retained. The flight control of the drone is not affected. It not only improves the flexibility of path planning, but also greatly guarantees the safety of MAV flight.

5. Experimental Results

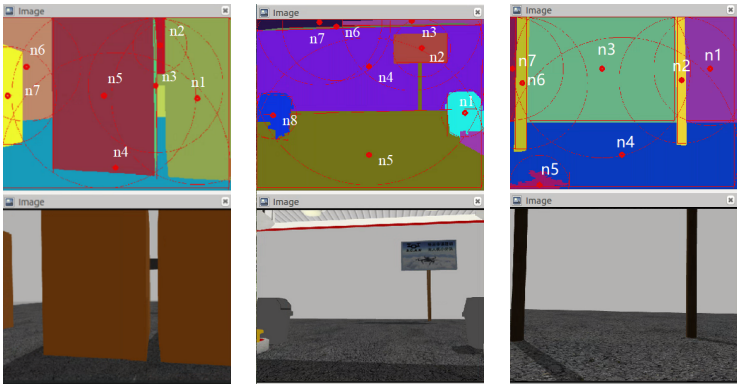
In this section, we present experimental results obtained using the proposed approach. We combined the drone simulation platform Rotors simulator [24] and the ROS software platform to build a drone autonomous navigation system simulation platform. We provided six kinds of simulation maps to test the performance of the planning algorithm and autonomous navigation system. The Rotors simulator provides a simulated RGB-D camera, which provides RGB and depth images similar to the real camera at 20 FPS, and the effective range of the depth image is set to 4.8 m, which is based on a general RGB-D camera as a reference standard. ROS is a well-known distributed operating system, which provides task scheduling for tasks in the form of nodes. The spatial-information extraction model we proposed is encapsulated as a Graph Cut node in the ROS system and runs at a frequency of 4 Hz, which continuously provides seeds for the intermediate goal. Planning is also packaged in the form of an Intermediate Goal Planning node in ROS and runs at a frequency of 2 Hz. Additionally, dt is set to extract control points at 0.5 s for optimization. Six simulation maps include three typical application scenarios: forests, urban building areas, and underground mines. Some holes are designed in the urban building area to test the ability of planning to penetrate and avoid obstacles.

5.1. Spatial-Structure Information Extraction

In the experiment, we set parameters $sigma = 0.5$, $k = 2000$, $Min = 2000$ to the graph-based image segmentation algorithm and checked the effect of spatial-structure information extraction method. Some of these images are obtained from the software Gazebo, simulating the actual environment, and the other part is obtained from the real environment. The image's resolution is 640×480. The result is shown in Figure 5. It can be seen that the center point extracted by the image in the simulation environment is very effective, and it can basically match the center of the object and the gap in the image. Due to the complex environment, the cut block became irregular, but different objects and gap centers can still be extracted. This verifies the applicability of our model to the actual environment.

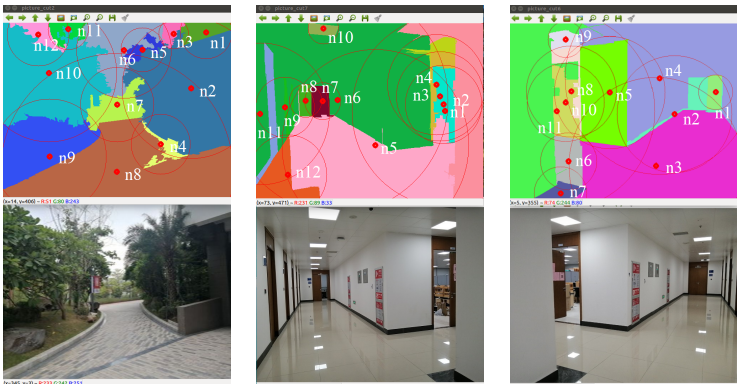
5.2. Across the Gap

In order to estimate the performance of the spatial structure extraction model, we used a pavilion with four holes as a test scene. The goal position was set at $(6.0, 0.0, 1.0)$, and the drone started from a distance of about 6 m from it. The directions are 0°, 15°, 30°, and 45°. The result is shown in Figure 6. It shows that the first intermediate goal is directly positioned to the geometric center of the hole at 0°, 15° and 30°. Then, the drone is guided to safely fly to the goal point. At the 45° position, the drone did not find the best intermediate goal under the initial search, but the geometric center of the hole was still located during the flight, so as to guide the drone to reach the goal position safely.



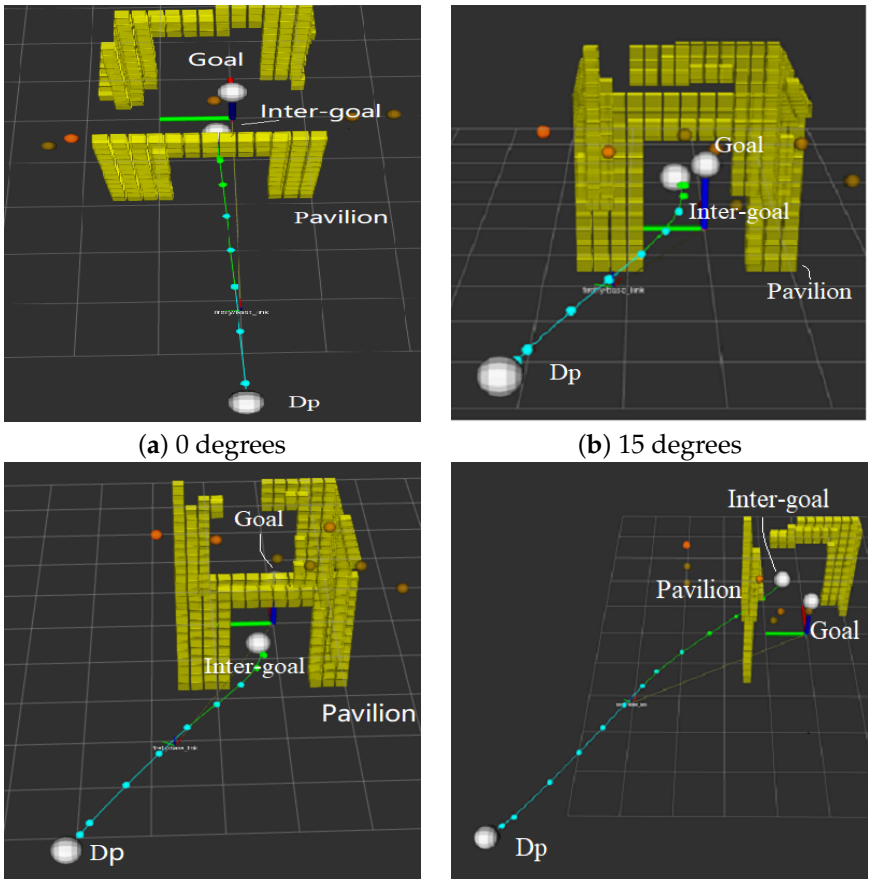
(**a**) simulation (**b**) simulation (**c**) simulation

Figure 5. *Cont.*



(**d**) forest (**e**) indoor (**f**) indoor

Figure 5. Segmented image. (**a**–**c**) are the images obtained in the simulation environment. (**d**–**f**) are the real environment images. The original image data are below, and the segmented blocks are on the top. The ellipse in the image is the smallest circumscribed ellipse of the block (red line). The viewpoint points to the geometric center of the object or the gap (red point).



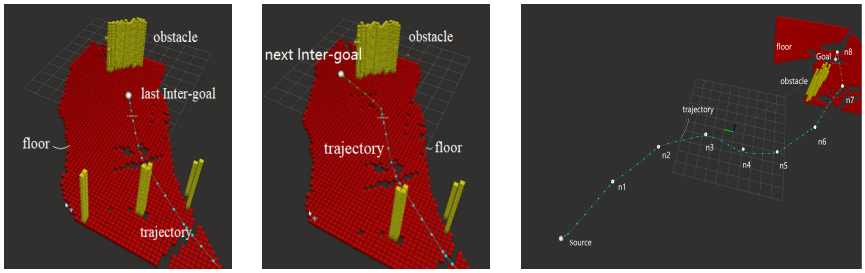
(**a**) 0 degrees (**b**) 15 degrees

(**c**) 30 degrees (**d**) 45 degrees

Figure 6. Planning A through the pavilion with four holes from different angles. (**a**–**d**) are 0 degrees, 15 degrees, 30 degrees and 45 degrees. Dp point is the drone position, inter-goal point is the intermediate goal, goal point is the goal. Pavilion and circular buffer map are marked.

5.3. Local Planning Optimization

We used a variety of simulation scenarios to test the performance of our local planning optimizer. Part of the trajectory is shown in Figure 7. In the process of testing, the current intermediate goal is blocked by obstacles, which negatively affects local planning, but a new intermediate goal is selected quickly to guide local planning. Trajectory avoids obstacles smoothly and the original path will not be affected. Compared with the polynomial trajectory expression proposed in [3], the time cost and smoothness are greatly improved. The smooth trajectory can greatly ensure the safety of MAV flight and effectively shorten the flight trajectory.



(**a**) Iterate Before (**b**) Iterate After (**c**) Global Trajectory

Figure 7. The floor (red grid) and obstacles (yellow grid) within circular buffer are shown, and the intermediate goal, source and goal are marked in (**a**–**c**). In (**c**), $n1$–$n8$ are all intermediate goal points. The global trajectory (blue line) is generated by the control points of B-spline (bule points).

5.4. System Simulation

In order to evaluate different aspects of our autonomous navigation system, we compare our system and Ewok [6] in different scenarios, and record the time cost, distance cost, and success ratio, respectively. Both our method and Ewok take the path planning of a fixed source point and goal point as the basic task. Ewok optimizes the global trajectory between the source point and the goal point to obtain a collision-free trajectory and also expresses the trajectory in B-spline. Our method combines the expression trajectory based on B-spline with the intermediate goal strategy and uses the intermediate goal to guide the local path optimizer to reach the goal point. Map No.1 is a forest with $15 \times 15 \times 5$ m^3, in which the brown cylinder represents the tree trunk and the gray represents the ground surface. Maps No.2, No.3 and No.4 add relatively large obstacles and irregular obstacles on the basis of map No.1 to build a scene that may cause the MAV to fall into a local optimal solution. The source point is set as $(-10.0, 10.0, 1.0)$. The goal point is set as $(10.0, -10.0, 1.0)$. Map No.5 simulates the environment of the building area with $10 \times 10 \times 5$ m^3, and No.6 simulates the scene of an underground mine with $10 \times 10 \times 5$ m^3. Map No.7 is a large-scale map with $50 \times 50 \times 5$ m^3, which is equipped with houses, vehicles, gas stations, etc. It is used to comprehensively test the performance of the algorithm in dealing with complex environments, as show in Figure 8. Each map is tested 25 times, and the data were obtained after averaging. The results of the simulations are shown in Tables 1 and 2.

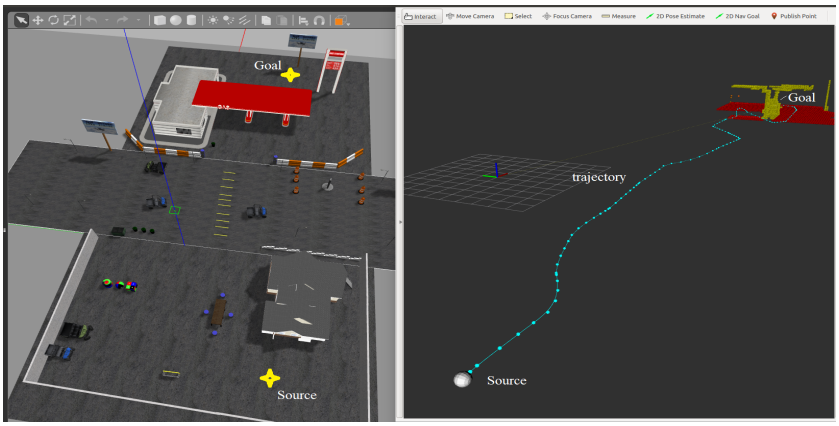


Figure 8. Large-scale simulation map and trajectory. In the simulation scene panel (**left** side), *Source* and *Goal* are marked (yellow asterisk). In the information display panel (**right** side), *Source* and *Goal* are marked (white balls).

Table 1. Simulation data record.

Evaluation Metric	Map 1	Map 2	Map 3	Map 4
Path length [m]	28.284	28.284	28.284	28.284
Ewok time [s]	25.470	25.668	28.995	/
Our method time [s]	32.130	36.581	39.410	36.209
Ewok trajectory length [m]	28.401	32.197	33.457	/
Our method trajectory length [m]	43.830	51.220	56.469	51.343
Ewok success ratio	100%	24%	8%	/
Our method success ratio	100%	88%	88%	92%

Table 2. Simulation data record.

Evaluation Metric	Map 5	Map 6	Map 7
Path length [m]	16.0	25.0	51.0
Our method time [s]	26.676	29.273	64.6065
Our method trajectory length [m]	35.178	38.914	81.9164
Our method success ratio	84%	84%	64%

The results show that Ewok reached the goal with a 100% success rate, its trajectory maintained smoothness and was collision-free, and it performed perfectly in terms of time and distance in map No.1. However, EWOK's success rate is greatly reduced when faced with large and irregular obstacles in maps No.2, No.3 and No.4. In its successful cases, the MAV frequently approaches the obstacle surface during the flight, leading to difficulty in ensuring the safety of drones. On the contrary, our algorithm can still maintain a high success rate in complex scenarios, while maintaining a suitable distance between multiple obstacles, which greatly guarantees the safety of the drone.

In maps No.5, No.6 and No.7, the MAV can only pass through the door to reach the goal due to the occlusion of the wall. EWOK cannot obtain a feasible path through re-planning optimization, but our intermediate goal strategy plays an important role. The position of the door can be found with the help of spatial-structure information and switching flight status. When the drone falls into a local optimal problem, our planning algorithm can quickly pull the drone out of the local optimal solution area. It can be seen that in a space with a more regular structure, the intermediate goal is more accurate, and the drone is safer.

6. Conclusions

This paper presents a set of vision-based autonomous navigation systems, which can still provide a collision-free and real-time trajectory in an environment with densely distributed obstacles, as well as with a previously unknown map. We use the basic idea of graph-based image segmentation to construct a spatial extraction model, combined with the exploration-inspired sampling method. We improved the optimization performance by adding the control point of B-Spline with an intermediate goal. The simulation results verified that our system can still show excellent performance in the face of complex and changeable scenarios in contrast to Ewok.

Author Contributions: Methodology, X.Z.; data collection, X.Z.; formal analysis, Z.Y. and X.Q.; writing—original draft preparation, X.Z., J.C. and Z.Y.; writing—review and editing, X.Z. and Z.Y.; supervision, J.C. and X.Q.; project administration Z.Y. All authors have read and agreed to the published version of the manuscript.

Funding: The National Natural Science Foundation of China (61871426), Guangdong Science and Technology Planning Project (2018B030322004) and The Natural Science Foundation of Shenzhen (ZX20210222), The Major Key Project of PCL (PCL2021A03-1).

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

Acknowledgments: The authors would like to express their appreciation to the support from the National Natural Science Foundation of China, Guangdong Science and Technology Planning Project and the Natural Science Foundation of Shenzhen.

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

References

1. Calisi, D.; Farinelli, A.; Iocchi, L.; Nardi, D. Autonomous exploration for search and rescue robots. In *Urban Transport XII: Urban Transport and the Environment in the 21st Century*; WIT Press: Southampton, UK, 2007; Volume 94, pp. 305–314.
2. Thrun, S.; Thayer, S.; Whittaker, W.; Baker, C.; Burgard, W.; Ferguson, D.; Hannel, D.; Montemerlo, M.; Morris, A.; Omohundro, Z.; et al. Autonomous exploration and mapping of abandoned mines. *IEEE Robot. Autom. Mag.* **2004**, *11*, 79–91. [CrossRef]
3. Oleynikova, H.; Taylor, Z.; Siegwart, R.; Nieto, J. Safe Local Exploration for Re-planning in Cluttered Unknown Environments for Microaerial Vehicles. *IEEE Robot. Autom. Lett.* **2018**, *3*, 1474–1481. [CrossRef]
4. Chen, J.; Liu, T.; Shen, S. Online generation of collision-free trajectories for quadrotor flight in unknown cluttered environments. In Proceedings of the 2016 IEEE International Conference on Robotics and Automation (ICRA), Stockholm, Sweden, 16–21 May 2016; pp. 1476–1483.
5. Alzugaray, I.; Teixeira, L.; Chli, M. Short-term UAV path-planning with monocular-inertial SLAM in the loop. In Proceedings of the 2017 IEEE International Conference on Robotics and Automation (ICRA), Singapore, 29 May–3 June 2017.
6. Usenko, V.; von Stumberg, L.; Pangercic, A.; Cremers, D. Real-time trajectory re-planning for MAVs using uniform B-splines and 3D circular buffer. In Proceedings of the 2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Vancouver, BC, Canada, 24–28 September 2017; pp. 215–222.
7. Karaman, S.; Frazzoli, E. Incremental Sampling-based Algorithms for Optimal Motion Planning. *Int. J. Robot. Res.* **2011**, *30*, 846–894. [CrossRef]
8. Florence, P.; Carter, J.; Tedrake, R. Integrated Perception and Control at High Speed: Evaluating Collision Avoidance Maneuvers without Maps. In *Algorithmic Foundations of Robotics XII*; Springer: Cham, Switzerland, 2016.
9. Lopez, B.T.; How, J.P. Aggressive 3-D collision avoidance for high-speed navigation. In Proceedings of the 2017 IEEE International Conference on Robotics and Automation (ICRA), Singapore, 29 May–3 June 2017; pp. 5759–5765.
10. Zucker, M.; Ratliff, N.; Dragan, A.D.; Pivtoraiko, M.; Klingensmith, M.; Dellin, C.M.; Bagnell, J.A.; Srinivasa, S.S. CHOMP: Covariant Hamiltonian optimization for motion planning. *Int. J. Robot. Res.* **2013**, *32*, 1164–1193. [CrossRef]
11. Oleynikova, H.; Burri, M.; Taylor, Z.; Nieto, J.; Siegwart, R.; Galceran, E. Continuous-time trajectory optimization for online UAV re-planning. In Proceedings of the 2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Daejeon, Korea, 9–14 October 2016; pp. 5332–5339.
12. LaValle, S.M. *Rapidly-Exploring Random Trees: A New Tool for Path Planning*; Technical Report; Computer Science Department, Iowa State University: Ames, IA, USA, 1998; pp. 98–111.
13. Bry, A.; Roy, N. Rapidly-exploring Random Belief Trees for motion planning under uncertainty. In Proceedings of the IEEE International Conference on Robotics and Automation (ICRA), Shanghai, China, 9–13 May 2011.

14. Bircher, A.; Kamel, M.; Alexis, K.; Oleynikova, H.; Siegwart, R. Receding horizon "next-best-view" planner for 3D exploration. In Proceedings of the 2016 IEEE International Conference on Robotics and Automation (ICRA), Stockholm, Sweden, 16–21 May 2016; pp. 1462–1468.
15. Dai, A.; Papatheodorou, S.; Funk, N.; Tzoumanikas, D.; Leutenegger, S. Fast Frontier-based Information-driven Autonomous Exploration with an MAV. In Proceedings of the 2020 IEEE International Conference on Robotics and Automation (ICRA), Paris, France, 31 May 2020; pp. 9570–9576.
16. Witting, C.; Fehr, M.; Behnemann, R.; Oleynikova, H.; Siegwart, R. History-aware autonomous exploration in confined environments using MAVs. In Proceedings of the 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Madrid, Spain, 1–5 October 2018; pp. 1–9.
17. Amanatides, J.; Woo, A. A fast voxel traversal algorithm for ray tracing. *Eurographics* **1987**, *87*, 3–10.
18. Hornung, A.; Wurm, K.M.; Bennewitz, M.; Stachniss, C.; Burgard, W. OctoMap: An efficient probabilistic 3D mapping framework based on octrees. *Auton. Robot.* **2013**, *34*, 189–206. [CrossRef]
19. Felzenszwalb, P.F.; Huttenlocher, D.P. Efficient Graph-Based Image Segmentation. *Int. J. Comput. Vis.* **2004**, *59*, 167–181. [CrossRef]
20. Burri, M.; Oleynikova, H.; Achtelik, M.W.; Siegwart, R. Real-time visual-inertial mapping, re-localization and planning onboard MAVs in unknown environments. In Proceedings of the 2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Hamburg, Germany, 28 September–2 October 2015.
21. Richter, C.; Bry, A.; Roy, N. Polynomial Trajectory Planning for Aggressive Quadrotor Flight in Dense Indoor Environments. In *Robotics Research*; Springer: Berlin/Heidelberg, Germany, 2016.
22. De Boor, C. On calculating with B-splines. *J. Approx. Theory* **1972**, *6*, 50–62. [CrossRef]
23. Cox, M.G. The Numerical Evaluation of B-Splines. *IMA J. Appl. Math.* **1972**, *10*, 134–149. [CrossRef]
24. Furrer, F.; Burri, M.; Achtelik, M.; Siegwart, R. Robot operating system (ROS). In *Studies in Computational Intelligence*; Springer: Cham, Switzerland, 2016.