sensors

MDPI

Article

Position Checking-Based Sampling Approach Combined with Attraction Point Local Optimization for Safe Flight of UAVs

Hai Zhu [1], Baoquan Li [1,*], Ruiyang Tong [1], Haolin Yin [1] and Canlin Zhu [2]

[1] School of Control Science and Engineering, Tiangong University, Tianjin 300387, China; zhai08669827@163.com (H.Z.); tong166159@163.com (R.T.); haolinyinws@gmail.com (H.Y.)
[2] School of Communications and Information Engineering, Nanjing University of Posts and Telecommunications, Nanjing 210003, China; canlzhu@163.com
* Correspondence: libq@tiangong.edu.cn; Tel.: +86-139-2023-5240

Abstract: Trading off the allocation of limited computational resources between front-end path generation and back-end trajectory optimization plays a key role in improving the efficiency of unmanned aerial vehicle (UAV) motion planning. In this paper, a sampling-based kinodynamic planning method that can reduce the computational cost as well as the risks of UAV flight is proposed. Firstly, an initial trajectory connecting the start and end points without considering obstacles is generated. Then, a spherical space is constructed around the topological vertices of the environment, based on the intersections of the trajectory with the obstacles. Next, some unnecessary sampling points, as well as node rewiring, are discarded by the designed position-checking strategy to minimize the computational cost and reduce the risks of UAV flight. Finally, in order to make the planning framework adaptable to complex scenarios, the strategies for selecting different attraction points according to the environment are designed, which further ensures the safe flight of the UAV while improving the success rate of the front-end trajectory. Simulations and real-world experiment comparisons are conducted on a vision-based platform to verify the performance of the proposed method.

Keywords: sampling-based planner; unmanned aerial vehicle; motion planning

check for updates

Citation: Zhu, H.; Li, B.; Tong, R.; Yin, H.; Zhu, C. Position Checking-Based Sampling Approach Combined with Attraction Point Local Optimization for Safe Flight of UAVs. *Sensors* **2024**, *24*, 2157. https://doi.org/10.3390/s24072157

Academic Editors: Oscar Reinoso García, Luis Payá and Helder Jesus Araújo

Received: 1 February 2024
Revised: 19 March 2024
Accepted: 25 March 2024
Published: 27 March 2024

Copyright: © 2024 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (https://creativecommons.org/licenses/by/4.0/).

1. Introduction

Nowadays, with advantages such as high mobility, flexibility, and easy operation, quadcopter UAVs are popularly used in a variety of complex and dangerous environments. Particularly in tunnel exploration and emergency rescue scenarios without global positioning system (GPS) signals, UAVs can only rely on their own cameras, radars, and other sensors to continue to perform tasks, so vision-based perception and navigation algorithms have been the focus of more and more researchers. The UAV system is mostly composed of the four essential modules: environment perception, state estimation, motion planning, and dynamic control [1,2]. A collision-free, smooth, and dynamically feasible trajectory guarantees flight safety, which is generated by a motion planning module in working scenarios. Motion planning is generally divided into two parts: front-end discrete path search and back-end continuous trajectory optimization, aiming at generating a reference trajectory that satisfies the above three basic conditions for the controller to track [3,4]. Since quadrotor UAVs with small fuselages have limited computational resources of on-board computers, a major research objective is how to fully utilize the limited resources to generate front-end paths and back-end trajectories. The trade-off between front-end and back-end computational resource allocation is an issue that needs to be considered in motion planning [5–7].

Consider transforming the problem of resource trade-offs into one of maximizing the use of information in the planning process. In other words, it can also be described as accelerating the trajectory optimization process by obtaining higher quality paths through

using more parameter information in the front-end path generation process. Therefore, if the front-end path generation takes into account dynamics of UAVs, the back-end trajectory optimization will effectively decrease its burden. Zhou et al. propose a kinodynamic path search method to find initial trajectories that are safe, dynamically feasible and require the shortest time in a discrete control space [8]. A lightweight yet effective topology-guided kinodynamic planner for the fast flight of a quadrotor with limited onboard computational resources is proposed in [9]. Firstly, the topology of the environment is constructed through start and end points; next, the dynamics are sampled at the periphery of topological structure; and finally, the reference trajectory is obtained through local optimization and adjustment, which largely reduces the number of sampling points. In order to avoid the case that some narrow slits in the environment may provide more optimal solutions, Ye et al., in [10], propose a sampling-based kinodynamic planning method combined with local optimization, but this increases the risk of UAV collisions.

Through the above research work, it can be found that the use of bias sampling can be used to detect the position of some UAVs where collisions are possible, discarding some unnecessary sampling points, which not only ensures the flight safety of UAVs but also reduces the cost of path generation. Therefore, this paper considers the design of safer trajectory generation strategies that will make full use of the information of the environment topology. It then incorporates gradient-based trajectory optimization methods into the local optimization process, meaning that the trajectory replanning is formulated as a nonlinear optimization problem, with trade-offs between smoothing, safety, and dynamic feasibility, which can easily deform infeasible trajectory segments into feasible ones with very high efficiency and low memory requirements. In this way, it will largely reduce the computational burden of back-end trajectory optimization and improve the robustness in complex environments.

In this paper, which employs information from a topology-guided graph [9], a kinodynamic planning algorithm combining a position checking-based sampling method and attraction point local optimization is proposed, which both ensures the safe flight of UAVs and reduces the computational cost to a certain extent. The main contributions are as follows:

1. With the topology construction of the environment, the initial trajectory connecting the start and end points does not take obstacles into account. A designed spherical space is formed by taking the lengths of the two points where the trajectory enters and leaves the obstacle as diameters, and the centers of the two points as spheres. In the node expansion process, a position-checking method that directly discards the sampled points in the designed sphere space is proposed.
2. In order to exploit the already available information, a reduction of unnecessary calculations in the rewiring process is proposed by judging whether the midpoint of line segment connecting the two points is in the sphere or not; if it is, this means that this branch does not need to be calculated.
3. Different attraction point generation strategies are designed according to the environment in the local optimization process, and then the gradient-based trajectory optimization is used to keep the trajectory away from the surface of obstacles, to ensure flight safety.

In this paper, Section 2 introduces some related research work. Section 3 describes methods for generating front-end trajectories using information about the environment topology constructed. Section 4 designs attractor generation strategies for local trajectory optimization. Section 5 conducts simulations and real-world experiments on the proposed method and analyzes the results. Section 6 concludes this article.

2. Related Work

2.1. Sampling-Based Planning

Search-based path generation schemes can find globally optimal solutions with sufficient computational resources and time. However, the magnitude of the space complexity

of the algorithm grows exponentially with increasing dimensionality [11–13]. Sampling-based motion planning methods do not need to explicitly construct obstacle regions but instead use sampling to guide the search of configuration space (C-space). A large amount of C-space does not need to be explored and less on-board computational resources are consumed. Therefore, sampling-based algorithms are generally superior to search-based algorithms for mobile robots like UAVs that perform motion planning in high-dimensional spaces. The classical sampling-based algorithms are mainly divided into two categories: probabilistic roadmap (PRM) [14] based on global batch sampling and incremental iterative rapid exploration of random trees (RRT) [15]. Both types of algorithms choose the strategy for connecting the sampled nodes to the nearest nodes in the graph or tree, which is a fast way to explore the whole space, but does not utilize the existing structural information already available in the graph or tree. existing structural information in the graph or tree. As the demands of application scenarios increase, more and more algorithms based on them have been proposed. For example, the informed RRT* algorithm [16], which employs elliptical sampling instead of globally uniform sampling, is an incremental sampling-based method for optimal path planning. In addition, Arslan et al. in [17] proposed the RRT# algorithm, which does not rewire nonpromising nodes and uses cyclic rewiring to propagate the information obtained from sampling to the rest of the tree, improving the efficiency of the algorithm to converge on the optimal solution. In addition, Figure 1 gives a schematic diagram of the path search of the above three sampling algorithms in the same environment, with the same start and end points. As shown in Figure 1, compared to the other two algorithms, the RRT algorithm has the longest path in general because it does not have asymptotic optimality and samples randomly in the whole space. Both RRT# and Informed RRT* are based on the improvement of the RRT* algorithm with asymptotic optimality, and the path lengths are generally not much different, with the exception of being shorter, when compared to RRT. Informed RRT*will draw an ellipsoid space (gray-yellow area) with the start and end points as focuses after the end point is found; we will now sample this space (blue dots), so you can see that most of the blue points are in the ellipsoid space.

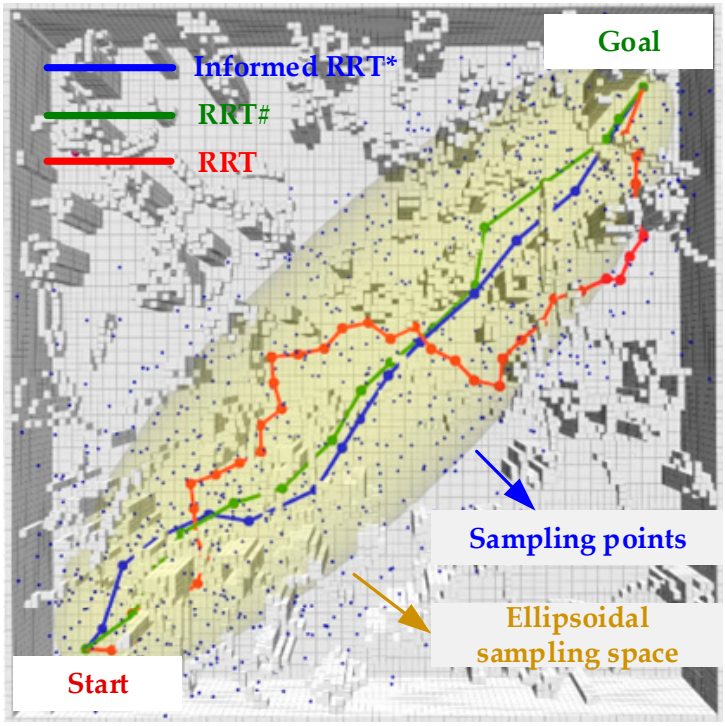


Figure 1. Diagram of the three sampling-based algorithms.

Boeuf et al. propose a quasi-metric to determine the proximity of a quadrotor and demonstrate that its integration into RRT-based and PRM-based algorithms reduces computational time by up to two orders of magnitude [18]. Karaman et al. show in [19] that the RRT* algorithm is an asymptotically optimal algorithm, which means that as time tends to infinity, the algorithm is able to find the optimal path. A sampling-based path search method, which allows sampling in regions where optimal solutions are more probable, is proposed in [20]. In addition, some sampling-based planning schemes incorporating neural networks are currently being proposed [21,22], which are effective in extracting the properties of high-dimensional features as a way of developing generalized methods to obtain probability distributions of samples, which do not only apply to specific scenarios and problems. However, it is difficult to obtain strong generalization ability with such methods, and the heavy network inference also makes it difficult to use these methods online in autonomous UAV navigation.

2.2. Kinodynamic Planning and Trajectory Optimization

The search-based kinodynamic planning scheme accomplishes path generation by using a discrete control space and searching for a segmented control solution through motion primitives. Liu et al. in [23] propose a search-based planning approach to find the smoothest and shortest time trajectories by exploring a map using a set of short time period motion primitives. Since sampling in state space, as in RRT*, tends to require solving the two-point boundary value problem (BVP) in optimal control, this means solving the equations of motion for a given dynamical system, given initial and termination conditions. This leads to inefficiency in expanding the tree in the high-dimensional state space, so this requires the design of a reasonable sampling strategy to minimize the unnecessary sampling points and accelerate the expansion of the tree, which will in turn improve the whole planning process.

Back-end trajectory optimization is essential to make the overall trajectory satisfy the dynamic demands of the whole system. The minimum snap trajectory generation algorithm proposed in [24] has been widely used in quadcopter UAV path planning efforts by solving a sequence of quadratic programming (QP) problems. Richter et al. in [25] prove that the minimum snap trajectories can be obtained by a closed-form solution, in which the trajectory is secured by iteratively adding intermediate track points. However, these general hard-constrained optimization schemes may cause a serious load on computational resources. Currently, the gradient-based soft constraint scheme is generally solved by modeling the trajectory optimization as a nonlinear optimization problem. In general, constructing a Euclidean signed distance field (ESDF) is necessary [26–28] to characterize constraints on safe obstacle avoidance, and provide information on the gradient of obstacle avoidance by linear interpolation in the ESDF, which has the advantage of being easy to compute but leads to the introduction of non-smooth functions. Of course, Zhou et al. propose a gradient-based planner without constructing an ESDF, which drastically reduces the amount of computation but also makes it easier for the UAV to fall into local optimality, and also degrades the trajectory quality [29]. In the motion planning process, trading off the resource allocation between the front and back ends also means minimizing the computational load of the on-board computer on the basis of ensuring the safety and feasibility of the reference trajectory.

3. Kinodynamic RRT* Planning Combined with Position Checking

3.1. Optimal State Transfer Cost

Based on the classic RRT* planning algorithm, the kinodynamic RRT* planning algorithm requires the calculation of state transfer cost between two nodes. Webb et al. in [30] construct a nilpotent dynamics matrix to find closed-form solutions for the two-point BVP, with the aim of saving computational cost. In addition, with the differential flatness property of a quadrotor UAV, state variables of the whole system can be represented by the four

flat outputs (position vector $\boldsymbol{p}(t) = [p_x(t), p_y(t), p_z(t)]^T$, yaw $\varphi(t)$) and their derivatives. Thus, state transition can be modeled as a linear system:

$$\dot{\boldsymbol{x}}(t) = \boldsymbol{A}\boldsymbol{x}(t) + \boldsymbol{B}\boldsymbol{u}(t) \tag{1}$$

where system states and control input with jerk are shown as follows:

$$\begin{array}{c} \boldsymbol{x}(t) = [\boldsymbol{p}(t)\ \dot{\boldsymbol{p}}(t)\ \ddot{\boldsymbol{p}}(t)]^T, \\ \boldsymbol{u}(t) = \dddot{\boldsymbol{p}}(t), \end{array} \boldsymbol{A} = \begin{bmatrix} \boldsymbol{0}_{6\times3} & \boldsymbol{I}_{6\times6} \\ \boldsymbol{0}_{3\times3} & \boldsymbol{0}_{3\times6} \end{bmatrix},\ B = \begin{bmatrix} \boldsymbol{0}_{6\times3} \\ \boldsymbol{I}_{3\times3} \end{bmatrix} \tag{2}$$

where $\boldsymbol{0}_{r\times l}$ denotes the zero matrix for r rows and l columns, and $\boldsymbol{I}_{e\times e}$ denotes the e-dimensional identity matrix.

Euclidean or Manhattan distance can be used for path planning if motion constraints are not considered. Optimal control is utilized for trajectory planning under motion constraints, and energy and time optimality are considered to design cost functions for state transition. If the transition cost between two states is small, they are affirmed to be close. Transition cost can be calculated based on arrival time τ and control input $\boldsymbol{u}(t)$, and the cost function from state $\boldsymbol{x}_0$ to state $\boldsymbol{x}_1$ is defined to be

$$J(\boldsymbol{x}_0, \boldsymbol{x}_1) = \int_0^\tau (\rho + \boldsymbol{u}(t)^T \boldsymbol{u}(t)/2) dt \tag{3}$$

where ρ is a weight for time and control cost. For searching a trajectory from the initial state $\boldsymbol{x}_{start}$ to the target state range space $\mathcal{X}_{goal}$, the following model is established with computing optimal arrival time:

$$\begin{array}{l} \min\limits_{x(t)} J = \int_0^\tau (\rho + \boldsymbol{u}(t)^T \boldsymbol{u}(t)/2) dt \\ s.t.\ \ \boldsymbol{A}\boldsymbol{x}(t) + \boldsymbol{B}\boldsymbol{u}(t) - \dot{\boldsymbol{x}}(t) = \boldsymbol{0}, \\ \quad \boldsymbol{x}(0) = \boldsymbol{x}_{start}, \boldsymbol{x}(\tau) \in \mathcal{X}_{goal}, \\ \quad \forall t \in [0, \tau], \boldsymbol{x}(t) \in \mathcal{X}^{free}, \boldsymbol{u}(t) \in \mathcal{U}^{free} \end{array} \tag{4}$$

where $\mathcal{U}^{free}$ denotes feasible control set, and $\mathcal{X}_{goal}$ satisfies derivative constraints and does not contain obstacles. Solving an optimal trajectory edge between two neighboring state points in this paper is equivalent to solving the two-point BVP. During the node tree extension, the state at the start moment $\boldsymbol{x}(0)$ is fully given, while the state at the termination moment $\boldsymbol{x}(\tau)$ is partially constrained. As for the node rewiring, the state at the starting moment $\boldsymbol{x}(0)$ and the state at the termination moment $\boldsymbol{x}(\tau)$ are known nodes in the tree, so the constraints at both ends are fully given. In summary, the trajectory solution problem can be described as an optimal control problem with free terminal moments and constrained terminal states. The optimal arrival time and polynomial coefficients can be solved according to the Pontryagin's maximum principle, so that the optimal trajectory $\boldsymbol{p}^*(t)$ as well as the optimal control inputs $\boldsymbol{u}^*(t)$ are obtained.

*3.2. Kinodynamic RRT**

The kinodynamic RRT* algorithm is a sampling algorithm for solving path planning problems with dynamics constraints. It is an extension of the conventional RRT* algorithm that is intended to make it applicable to systems with continuous state and control input spaces, which is systematically proposed in [30], as shown in Algorithm 1.

Initially, the start point is used as the root node of the tree and an empty search tree T is initialized. Then a state $\boldsymbol{x}_{new}$ is randomly sampled through RandomlySample $(\boldsymbol{x}_i, E)$, with position, velocity, and acceleration information obtained within a set number of samples and sampling time, which must satisfy dynamics constraints of the system. Next, FindNeighbor $(\boldsymbol{x}_{new}, T)$ and Expansion $(\boldsymbol{x}_{new}, \mathcal{X}_{nearest}^{backward})$ are used to find a state $\boldsymbol{x}_{nearest}^{backward}$ around state $\boldsymbol{x}_{new}$ that has the smallest transfer cost and serves as the parent node of $\boldsymbol{x}_{new}$. Here, a check is made to see if the target region has been reached; if so, no further

operations are needed and the searched tree T is returned. Instead, branches are rewritten using FinNeighbor $(\boldsymbol{x}_{new}, T)$ and E, which makes the resulting path asymptotically optimal. A gradual optimal path is gradually obtained through the above cycle of node tree extension as well as node rewiring. When the stopping condition of the loop is reached (e.g., the number of nodes in the trajectory tree reaches the set maximum value or a sampling time threshold is exceeded). Moreover, Figure 2 shows a demonstration case of the RRT* algorithm that considers the dynamics. As shown in Figure 2, the kinodynamic RRT* algorithm randomly samples in an environment with randomly distributed obstacles (light blue circles and bars). In this case, the green curve is the trajectory connecting the sampling points (red dots) and the start point; when the set goal (blue squares) is sampled, a red trajectory connecting the start and end points is generated, and the algorithm will terminate its operation.

Algorithm 1: Classical Kinodynamic RRT*

Notation: Environment E, Tree T, Sampling State $\boldsymbol{x}$, $\boldsymbol{x}_{start} \in \mathcal{X}^{free}$, $\boldsymbol{x}_{goal} \in \mathcal{X}^{free}$, Maximum Sampling Number n
$T \leftarrow \{\boldsymbol{x}_{start}\}$
for $i \in \{1, n\}$ **do**
  $\boldsymbol{x}_{new} \leftarrow$ RandomlySample $(\boldsymbol{x}_i, E)$;
  $\mathcal{X}^{backward}_{nearest} \leftarrow$ FindNeighbor $(\boldsymbol{x}_{new}, T)$;
  if $(\mathcal{X}^{backward}_{nearest} \neq \varnothing)$ **then**
    $\left\{\boldsymbol{x}^{backward}_{nearest}, \boldsymbol{x}_{new}\right\} \leftarrow$ Expansion $(\boldsymbol{x}_{new}, \mathcal{X}^{backward}_{nearest})$;
    $T \leftarrow$ TreeUpdate $(\left\{\boldsymbol{x}^{backward}_{nearest}, \boldsymbol{x}_{new}\right\}, T)$;
      if (getGoalRegional $(\boldsymbol{x}_{new}, \boldsymbol{x}_{goal})$) **then**
        break;
      end if
      $\mathcal{X}^{forward}_{nearest} \leftarrow$ FindNeighbor $(\boldsymbol{x}_{new}, T)$;
      if $(\mathcal{X}^{forward}_{nearest} \neq \varnothing)$ **then**
        Rewire $(\mathcal{X}^{forward}_{nearest}, T)$;
      end if
  end if
  return T
end for

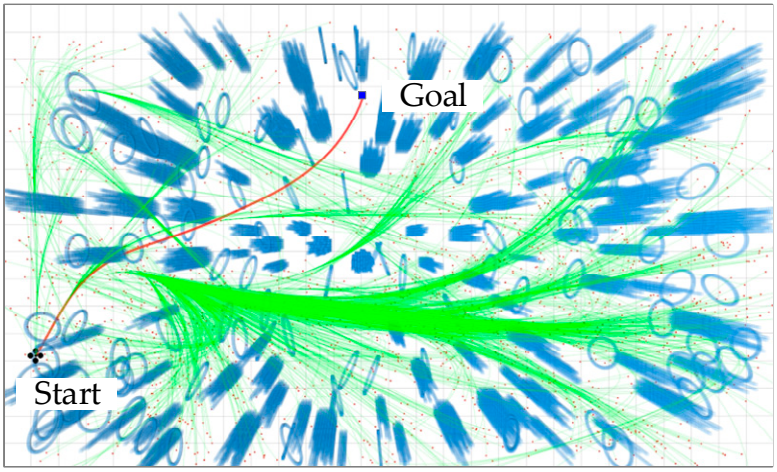


Figure 2. A snapshot of the demonstration of the kinodynamic RRT* algorithm.

In general, the two core steps of the RRT* algorithm, node tree extension and node rewiring are shown in Figure 3a–c below. In this case, the black circles indicate the nodes with numbers, Node 0 is the start node, and Node 5 is the nearest neighbor node $\boldsymbol{x}_{near}$ of

the new sampling point x_{new}; and the numbers on the path connecting the two nodes are the cost to be spent, which is generally obtained by solving the BVP if the UAV state is not considered. The space surrounded by the green circle drawn in dashed lines is the range of the parent node selected for the x_{new}. As shown in Figure 3a, since the cost of the path from Node 0 to x_{new} via Node 6 is less than that of path to x_{new} via any other node in the above range, Node 6 is set as the parent node of the new node, as shown by the green path in Figure 3b. In addition, as shown in Figure 3c, x_{new} is set as the parent of the surrounding nodes, and compared to Node 6, the cost of the path of setting x_{new} as the parent of Node 7 will become smaller, which means that the more optimal path is Node 0 to Node 7 via Node 6 and x_{new}, which also demonstrates the asymptotic optimality of the RRT* algorithm.

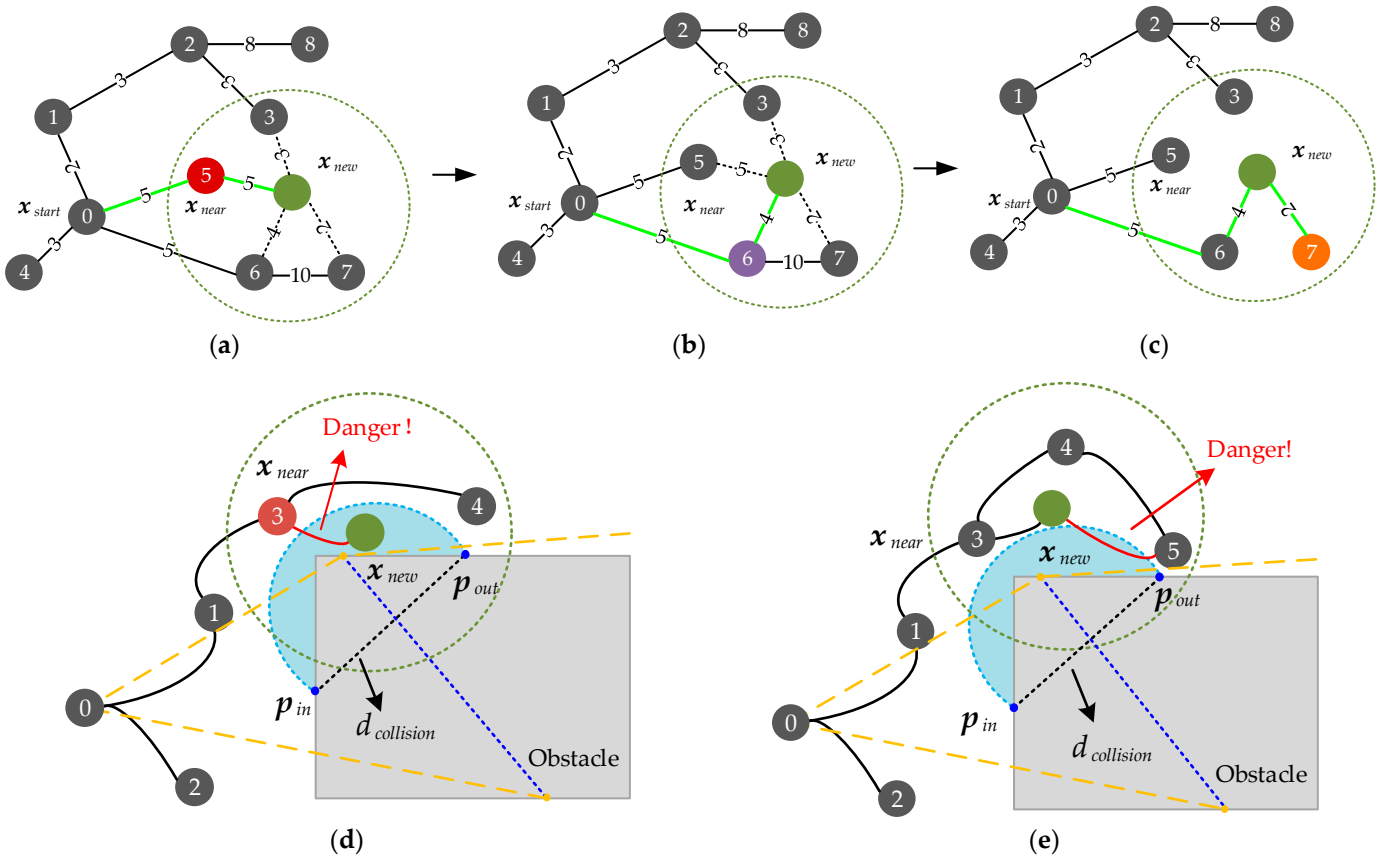


Figure 3. Diagram of the node expansion and node reconnection phases of the RRT* algorithm and motivation for node position checking in combination with the topology of the environment. (**a**) Find the lowest cost parent among neighbor nodes; (**b**) The new sampling point is reconnected to the lowest-cost parent node; (**c**) Use the new sampling point as a parent node to find less costly a path; (**d**) Motivation for adding sample point position checking to the node tree expansion process; (**e**) Motivation for adding sample point position checking to the node rewiring process.

In this paper, through the above analysis of the two key steps of the RRT* algorithm, as shown by the orange dotted line in Figure 3d,e, the topology-guided graph [9] is constructed by using the two points $\{p_{in}, p_{out}\}$ of initial trajectory through the obstacle and their perpendicular bisector (blue dashed line). The sampling guided by the guided graph of the environment will make the trajectory generated around $\{p_{in}, p_{out}\}$ bring a high risk to the flight of the UAV. Therefore, in the node tree expansion as well as node rewiring stage, this paper exploits the information of the topology-guided graph to propose the strategy of dividing a spherical space (light blue area) with diameter size $d_{collision}$. It does this to check the position of sampling points, which not only reduces unnecessary BVP solving but also moves the trajectory far away from obstacles and ensures the safety of UAV

flight. In particular, the spherical space construction method and the position-checking strategies are described in detail below.

3.3. Node Tree Extension Optimization

Considering that, as mentioned above, establishing a connecting trajectory between two states (namely, solving the two-point BVP between state nodes) consumes a large number of computational resources, the method of sampling the environment by roughly capturing the topology of the environment and guiding sampling around the periphery of the topology has been proposed. Sampling around the planar intersections of the topology of the environment can cause the problem of being too close to obstacles. Therefore, this paper considered utilizing the information of environment topology for position checking of the new sampling points, which not only reduces some ineffective sampling points but also ensures the safety of generated trajectories by expanding the sampling range.

Firstly, as shown in Figure 4, a red initial trajectory connecting the start and end points is generated by solving the two-point BVP without considering obstacles. Next, obstacle checking is performed to obtain two points $\{p_{in}, p_{out}\}$ where the initial trajectory enters and leaves the obstacle. Then, the perpendicular bisector (blue dashed line) of the line connecting the two points is extended outward into the collision-free space and used as the vertices of the topology. In this way, a topology-guided graph that gives a rough indication of the environmental information is constructed. Finally, biased sampling is performed based on the constructed environment structure to improve the sampling efficiency.

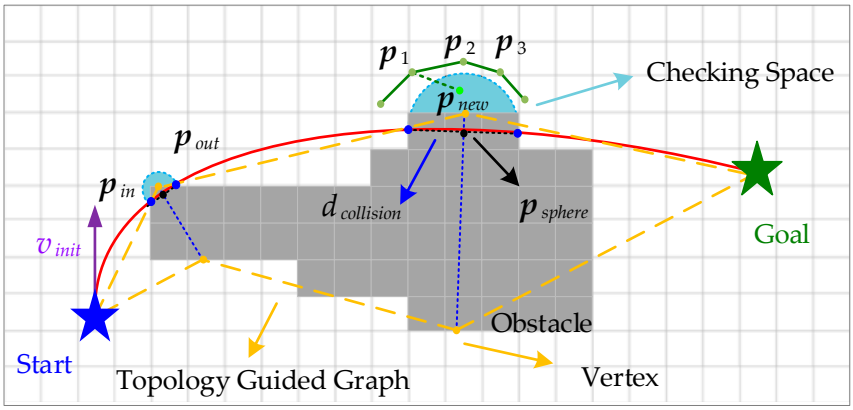


Figure 4. Expand tree nodes around vertices. Consider whether to add to the tree T by performing a position check on the new node.

In order to ensure that the sampling around the vertices can be as far away from the obstacles as possible, since $\{p_{in}, p_{out}\}$ is closely related to them, it is considered to use the distance between two points to delimit a spherical checking space, whereby sampling points within this space cannot pass the position check. Specifically, position vector p_{new} in the new sampling point x_{new} is utilized to check:

$$\begin{aligned} d_{collision} &= \|p_{out} - p_{in}\|_2, \\ p_{sphere} &= (p_{out} - p_{in})/2, \\ \Delta d_{new} &= \left\|p_{new} - p_{sphere}\right\|_2 \end{aligned} \tag{5}$$

where $d_{collision}$ is the diameter of the sphere, p_{sphere} is the center of the sphere, and Δd_{new} is the distance from the sampling point to the center of the sphere. If $\Delta d_{new} < d_{collision}/2$, which indicates that the sampling point is in the middle of the sphere space, this point is discarded.

This process is performed in Expansion $(x_{new}, \mathcal{X}_{nearest}^{backward})$ in Algorithm 1, which ensures that other optimal or suboptimal sampling points are not affected and can be discarded

before the two-point BVP is solved. In contrast to the algorithm in [10], a spherical space divided around the vertices of the topology-guided graph is proposed in the paper to perform a position check on the newly sampled points. This additional position checking not only reduces the number of triggers for local optimization of the back-end trajectory but also makes the flight safer.

3.4. Rewire Optimization

Unlike the RRT algorithm, the local optimality of the RRT* algorithm is safeguarded by the fact that it has a reconnection component that ensures its presence every time a new branch is successfully added to the node tree. This new sampling point is set to be the parent of a certain range of nodes in the surrounding area, and then the lower cost branch is selected by the cost change. However, the computational cost of the BVP is large in sampling-based planners, and thus some unnecessary solving needs to be avoided.

As described in Section 3.2, if motion constraints are not considered, the Euclidean distance or the Manhattan distance can be used. In contrast, in a state space with motion constraints, the introduction of optimal control is considered. Therefore, as shown in Figure 5, consideration can be given to make full use of the sphere with diameter $d_{collision}$, which is solved in the above section by assuming that nodes $\boldsymbol{p}_a, \boldsymbol{p}_b, \boldsymbol{p}_c$ exist around the newly added node $\boldsymbol{x}_{new}$ in the tree, and next treating $\boldsymbol{x}_{new}$ as the parent of the surrounding nodes, before judging it by Rewire $(\mathcal{X}_{nearest}^{forward}, T)$ against $\left\{\boldsymbol{p}_{new}, \boldsymbol{p}_j\right\}, j = a, b, c$, before solving the BVP by setting the judging condition as

$$\begin{aligned} \boldsymbol{p}_{center} &= (\boldsymbol{p}_{new} - \boldsymbol{p}_c)/2, \\ \Delta d_{center} &= \left\| \boldsymbol{p}_{sphere} - \boldsymbol{p}_{center} \right\|_2 \end{aligned} \tag{6}$$

where $\boldsymbol{p}_{center}$ is the midpoint between the parent node $\boldsymbol{p}_{new}$ and child node $\boldsymbol{p}_c$. Δd is the distance from the midpoint to the center of the sphere. If $\Delta d_{center} < d_{collision}/2$, it shows that the reconnection of these two nodes is not necessary. Therefore, by checking the position of the midpoints of the above connecting lines in the rewiring phase, there will be less meaningless BVP solving, which also prevents the generation of high-risk reference trajectories. Algorithm 2 shows the implementation process of node tree expansion and reconnection.

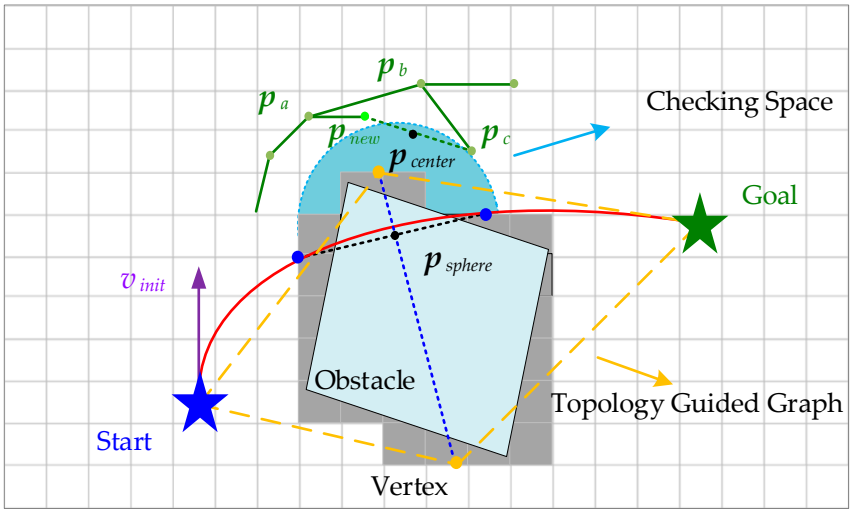


Figure 5. Considering the position-checked rewire to reduce the unnecessary two-point BVP solving process.

Algorithm 2: Kinodynamic RRT* with Position Check

Notation: Environment E, Tree T, Sampling State $\boldsymbol{x}$, $\boldsymbol{x}_{start} \in \mathcal{X}^{free}$, $\boldsymbol{x}_{goal} \in \mathcal{X}^{free}$, Maximum Sampling Number n, $\{\boldsymbol{p}_{in}, \boldsymbol{p}_{out}\}$
$T \leftarrow \{\boldsymbol{x}_{start}\}$
for $i \in \{1, n\}$ **do**
  $\boldsymbol{x}_{new} \leftarrow$ RandomlySample $(\boldsymbol{x}_i, E)$;
  $\mathcal{X}_{nearest}^{backward} \leftarrow$ FindNeighbor $(\boldsymbol{x}_{new}, T)$;
  if $(\mathcal{X}_{nearest}^{backward} \neq \varnothing)$ **then**
    if $(\{\boldsymbol{p}_{in}, \boldsymbol{p}_{out}\} \neq \varnothing)$ **then**
      $d_{collision} \leftarrow$ ComputValue **()**;
      $\boldsymbol{p}_{sphere} \leftarrow$ ComputVector **()**;
      $\Delta d_{new} \leftarrow$ ComputValue **()**;
      if $(\Delta d_{new} < d_{collision}/2)$ **then**
        continue;
      end if
    end if
    $\left\{\boldsymbol{x}_{nearest}^{backward}, \boldsymbol{x}_{new}\right\} \leftarrow$ Expansion $(\boldsymbol{x}_{new}, \mathcal{X}_{nearest}^{backward})$;
  end if
  $T \leftarrow$ TreeUpdate $(\left\{\boldsymbol{x}_{nearest}^{backward}, \boldsymbol{x}_{new}\right\}, T)$;
  if getGoalRegional $(\boldsymbol{x}_{new}, \boldsymbol{x}_{goal})$ **then**
    break;
  end if
  $\mathcal{X}_{nearest}^{forward} \leftarrow$ FindNeighbor $(\boldsymbol{x}_{new}, T)$;
  while $((\mathcal{X}_{nearest}^{forward} \neq \varnothing))$ **do**
    if $(\{\boldsymbol{p}_{in}, \boldsymbol{p}_{out}\} \neq \varnothing)$ **then**
      $\boldsymbol{p}_{center} \leftarrow$ ComputVector **()**;
      $\Delta d_{center} \leftarrow$ ComputValue **()**;
      if $(\Delta d_{center} < d_{collision}/2)$ **then**
      Continue;
      end if
    end if
    Rewire $(\mathcal{X}_{nearest}^{forward}, T)$;
  end while
end for
return T

4. Local Trajectory Optimization

In the above section, the segmented fifth degree polynomial trajectory has been obtained by solving the optimal control problem. It is easy to see from the above Figures 4 and 5 that, since the constructed environment is very rough, relying only on this original trajectory for generating the final reference trajectory will result in a degradation of the quality of the generated trajectory. Therefore, a local optimization triggered by collision detection is proposed in [10] where, instead of direct trajectory discarding when a collision occurs in the generated trajectory, the failing trajectory is reused and locally adjusted by a gradient-based optimization scheme.

4.1. Optimization Problem Modeling

For systems with differential flatness property, each flat output dimension can be decoupled independently for planning. In particular, for a quadrotor UAV system, the state of the whole system can be represented by the four flat outputs (position: p_x, p_y, p_z, yaw angle: φ) and their derivatives as state variables, which are not considered in this paper with respect to their yaw angle φ.

A complex trajectory often needs to be represented by more than one polynomial in each dimension in space, and it is possible to divide the complete trajectory into multiple

segments according to time t. Suppose the i-th segment of a trajectory $p_i(t)$ expressed by k segments, n-th degree polynomial is

$$p_i(t) = c_{k,0} + c_{k,1}t + c_{k,2}t^2 + \cdots + c_{k,n}t^n = \boldsymbol{c}_i^T \boldsymbol{t}, \tag{7}$$

where $\boldsymbol{c}_i \in \mathbb{R}^{n+1}$ denotes the coefficients of the i-th trajectory, $\boldsymbol{t} = (1, t, t^2 \ldots, t^n)^T$, and $t \in [T_{i-1}, T_i]$. The objective of the optimization is to find the optimal coefficients for each trajectory. In this paper, the coefficient c_i of each segment of the trajectory is chosen to be solved to minimize the total cost of jerk and to satisfy the constraints, hence this is an optimization problem with constraints.

The triple integrator model, with jerk as the input, will constrain the position, velocity and acceleration of the head and tail of the trajectory $\boldsymbol{p}(t)$, and hence six equations are needed for constraints. Specifically, $\boldsymbol{p}_i(t)$ has six coefficients when $n = 5$, and then $\boldsymbol{p}(t)$ can be expressed completely by Equation (7), as

$$\boldsymbol{p}(t) = \begin{cases} c_{1,0} + c_{1,1}t + \cdots + c_{1,4}t^4 + c_{1,5}t^5, T_0 \le t \le T_1, \\ c_{2,0} + c_{2,1}t + \cdots + c_{2,4}t^4 + c_{2,5}t^5, T_1 \le t \le T_2, \\ \qquad \vdots \\ c_{k,0} + c_{k,1}t + \cdots + c_{k,4}t^4 + c_{k,5}t^5, T_{k-1} \le t \le T_k \end{cases} \tag{8}$$

where $t \in [T_0, T_k]$ denotes the time period of the whole trajectory.

Therefore, the objective function $Jerk(t)$ can be determined as

$$Jerk(t) = \boldsymbol{p}^{(3)}(t) = [0\ 0\ 0\ 6\ 24t\ 60t^2]c, \tag{9}$$

where $c = \left[c_1^T\ c_2^T \cdots c_k^T\right]^T = [c_{1,0} \cdots c_{1,5}; c_{2,0} \cdots c_{2,5};\ \cdots\ ]^T$. In order to minimize it, meaning to minimize the integral of $Jerk(t)$ over the whole time, it is usual to choose the 2-Norm for the solution, which can be converted to

$$\min_c Jerk(t) = \min_c \sum_{i=1}^{k} \int_{T_{i-1}}^{T_i} \left(\boldsymbol{p}^{(3)}(t)\right)^2 dt. \tag{10}$$

By setting $a = [0\ 0\ 0\ 6\ 24t\ 60t^2]^T$, the transformed objective function is obtained

$$\left(\boldsymbol{p}^{(3)}(t)\right)^2 = \left(\boldsymbol{a}^T\boldsymbol{c}\right)^T(\boldsymbol{c}\boldsymbol{a}) = \boldsymbol{c}^T\boldsymbol{a}\boldsymbol{a}^T\boldsymbol{c}. \tag{11}$$

Next, let $\boldsymbol{A} = \boldsymbol{a}\boldsymbol{a}^T$, $\int_{T_{i-1}}^{T_i} \left(\boldsymbol{p}^{(3)}(t)\right)^2 dt$ in the objective function be converted to

$$\int_{T_{i-1}}^{T_i} \left(\boldsymbol{p}^{(3)}(t)\right)^2 dt = \int_{T_{i-1}}^{T_i} \boldsymbol{c}^T\boldsymbol{A}(t)\boldsymbol{c}dt = \boldsymbol{c}^T \int_{T_{i-1}}^{T_i} \boldsymbol{A}(t)dt\boldsymbol{c}. \tag{12}$$

Calculating the integral part of this separately, a positive definite matrix Q_i is obtained:

$$Q_i = \int_{T_{i-1}}^{T_i} \boldsymbol{A}(t)dt = \begin{bmatrix} \boldsymbol{0}_{3\times3} & \boldsymbol{0}_{3\times3} \\ & 36t & 72t^2 & 120t^3 \\ \boldsymbol{0}_{3\times3} & 72t^2 & 192t^3 & 360t^4 \\ & 120t^3 & 360t^4 & 720t^5 \end{bmatrix}_{T_{i-1}}^{T_i}. \tag{13}$$

As a result, the optimization problem is converted into

$$\begin{aligned} Jerk(t) &= \sum_{i=1}^{k} \int_{T_{i-1}}^{T_i} \left(\boldsymbol{p}^{(3)}(t)\right)^2 dt = \sum_{i=1}^{k} c_i^T Q_i c_i \\ &= \begin{bmatrix} c_1^T & c_2^T & \cdots & c_k^T \end{bmatrix} \begin{bmatrix} Q_1 & & & \\ & Q_2 & & \\ & & \ddots & \\ & & & Q_k \end{bmatrix} \begin{bmatrix} c_1 \\ c_2 \\ \vdots \\ c_k \end{bmatrix} = \boldsymbol{c}^T \boldsymbol{Q} \boldsymbol{c}. \end{aligned} \tag{14}$$

Since the objective function is a quadratic form, the trajectory optimization problem can be transformed into a QP problem. Where the objective function is of quadratic form and the matrix of the quadratic terms is positive definite, a closed-form solution is enabled to keep the trajectory away from obstacles by designing the objective term of the collision avoidance cost. A number of schemes have been proposed on this basis to make the resulting UAV trajectory converge on being smooth and collision free, and not exceeding the velocity constraints [31,32].

4.2. Local Optimization Strategy Adjustment

As shown in Figures 4 and 5, the front-end trajectory tends to converge on the safe space based on discarding some sampling points and reducing the number of times of solving the BVP. However, in some complicated scenarios, such as very irregular obstacles, especially around $\boldsymbol{p}_{in}$ and $\boldsymbol{p}_{out}$, the state sampling points are close to obstacles, which results in the back-end optimized trajectory, bringing the risk of collision with obstacles to the UAV. In this paper, the quadratic objective function in Equation (14) consists of a smoothing cost term J_s, a trajectory position error cost term J_p, and a collision cost term J_c in each dimensional space, and thus the optimization problem is modeled as

$$\min J = \lambda_s J_s + \lambda_p J_p + \lambda_c J_c, \tag{15}$$

where $\lambda_s, \lambda_p, \lambda_c$ are corresponding weight parameters.

In the local optimization, to avoid confusion with $\{\boldsymbol{p}_{in}, \boldsymbol{p}_{out}\}$ above, for a collision trajectory edge, the collision start and collision end points are denoted respectively as $\{\boldsymbol{p}_{start}, \boldsymbol{p}_{end}\}$. When $\{\boldsymbol{p}_{start}, \boldsymbol{p}_{end}\}$ exists, the A* algorithm, a heuristic search algorithm for low-cost path, is utilized to search for a path Φ_{A*} from $\boldsymbol{p}_{start}$ to $\boldsymbol{p}_{end}$ on the surface of the obstacle, and a direction vector is used to guide the trajectory away from the obstacle by using the path points of Φ_{A*} and the center of the sphere $\boldsymbol{p}'_{sphere}$, similar to the one constructed in Section 3.3. The direction vector is used to guide the trajectory away from the obstacle. Due to the irregularity of the obstacles, it is not reasonable to consider only the midpoint $\boldsymbol{p}_m$ of Φ_{A*}, as this does not take into account the different levels of risk around points $\boldsymbol{p}_{start}$ and $\boldsymbol{p}_{end}$. Therefore, more refined attraction point $\boldsymbol{p}_{att}$ generation strategies need to be designed.

Specifically, an intuitive demonstration is shown in Figure 6, where the path points of Φ_{A*} are divided into two parts, Φ_{A*}^{start} and Φ_{A*}^{end}, by the midpoint $\boldsymbol{p}_m$ of Φ_{A*}, and then the midpoints $\boldsymbol{p}_m^{start}$ and $\boldsymbol{p}_m^{end}$ of the two parts are obtained, respectively. Next, the distance $d_{attBase}, attbase \in (start, m, end)$ from these three points $\boldsymbol{p}_m, \boldsymbol{p}_m^{start}, \boldsymbol{p}_m^{end}$ to the center of the sphere are computed, similarly to Equation (5). In the upper two cases in Figure 6, $\boldsymbol{p}_{att}$ is obtained by

$$\boldsymbol{p}_{att} = \max(d_{attBase} + d_{\min Thread}) \tag{16}$$

where $d_{\min Thread}$ is a set minimum value to prevent A* path points from being too close to obstacles and reducing the effectiveness of local optimization. Notably, the blue dashed line is used to visualize the maximum value and has no practical engineering significance.

In the third case, $\boldsymbol{p}_{att}$ is determined according to the intersection of the red dashed line connecting the extension points of d_{start}, d_{end} with the extension line of d_m. In addition, similar to the method in Section 3.3, the position of the selected attraction points is checked against the spherical space; if $\left\| \boldsymbol{p}_{att} - \boldsymbol{p}'_{sphere} \right\|_2 < d'_{collision}$, it means that no local

optimization is needed, and the trajectory is directly resolved. Instead, the collision cost term is designed in a style similar to Equation (14):

$$\begin{aligned} J_c &= \sum_{C_{att}} \int_{t_{s,att}}^{t_{e,att}} [\boldsymbol{p}(t) - (\max(d_{attBase}) + d_{\min Thread})]^2 dt \\ &= \sum_{C_{att}} \sum_{i \in attSeg} (\boldsymbol{c}_i - \boldsymbol{c}_i^{att})^T \int_{t_{s,i,attP}}^{t_{e,i,attP}} \boldsymbol{t}\boldsymbol{t}^T dt (\boldsymbol{c}_i - \boldsymbol{c}_i^{att}) \\ &= \sum_{C_{att}} (\boldsymbol{c} - \boldsymbol{c}^{att})^T \boldsymbol{Q}_{c,att} (\boldsymbol{c} - \boldsymbol{c}^{att}) \end{aligned} \tag{17}$$

where C_{att} denotes the set of attractive points, and $attSeg$ denotes the trajectory of the corresponding segment. Different attraction points are selected by the difference of environment, meaning that therefore the algorithm proposed in this paper is able to achieve better results in cluttered environments, as compared to [10]. In addition, Algorithm 3 gives the detailed process.

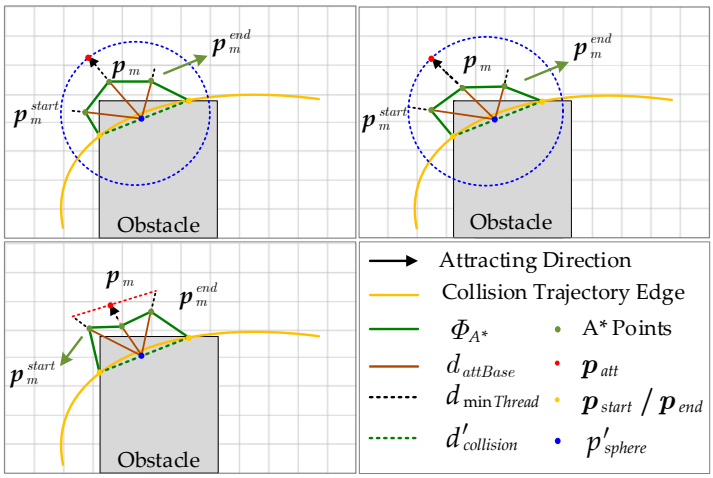


Figure 6. Diagram of the selection strategy of attraction points $\boldsymbol{p}_{att}$ in the process of regional trajectory optimization.

Algorithm 3: Attraction Points Selected by Environment

Notation: Environment E, $\boldsymbol{p}_{start}$, $\boldsymbol{p}_{end}$, $\boldsymbol{p}'_{sphere}$, $\boldsymbol{p}_{att}$, $d_{\min Thread}$
while$(\{\boldsymbol{p}_{start}, \boldsymbol{p}_{end}\} \neq \varnothing)$ **do**
  $\Phi_{A*} \leftarrow$ Search $(\boldsymbol{p}_{start}, \boldsymbol{p}_{end}, E)$;
  $\boldsymbol{p}_m \leftarrow$ GetMidpoint $(\boldsymbol{p}_{start}, \boldsymbol{p}_{end})$;
  $\left\{\Phi_{A*}^{start}, \Phi_{A*}^{end}\right\} \leftarrow$ DividePath $(\Phi_{A*}, \boldsymbol{p}_{start}, \boldsymbol{p}_m, \boldsymbol{p}_{end})$;
  $(\boldsymbol{p}_m^{start}, \boldsymbol{p}_m^{end}) \leftarrow$ EachMidpoint $(\boldsymbol{p}_m, \left\{\Phi_{A*}^{start}, \Phi_{A*}^{end}\right\})$;
  $d_{attBase} \leftarrow \left\|\boldsymbol{p}_{attBase} - \boldsymbol{p}'_{sphere}\right\|_2$;
  $d'_{attBase} \leftarrow d_{attBase} + d_{\min Thread}$;
  if $(d'_m \geq d'_{start} \parallel d'_m \geq d'_{end})$ **then**
    $\boldsymbol{p}_{att} \leftarrow$ GetMax $(d'_m, d'_{start}, d'_{end})$;
  else then
    $\boldsymbol{p}_{att} \leftarrow$ GetIntersection $(d'_m, d'_{start}, d'_{end})$;
    $d \leftarrow \left\|\boldsymbol{p}_{att} - \boldsymbol{p}'_{sphere}\right\|_2$;
    if $(d > d'_{collision})$ **then**
      return $\boldsymbol{p}_{att}$;
    else then
      break;
    end if
  end if
end while

5. Simulation and Experiment Results

5.1. Simulation

Simulation tests are run on a laptop with Intel i5-10200H processor, 8 GB RAM, 512 GB SSD and operating system Ubuntu 18.04 with melodic version of Robotics Operating System (ROS). In this case, the thresholds of velocity and acceleration are set to 4.0 m/s and 4.0 m/s^2, respectively, and the weighting factor in Equation (3) is set to 100.

Simulation 1: Feasibility verification. The whole planning system is tested and verified in a cluttered environment with an area of 50 m × 50 m × 5 m and 400 obstacles. In this case, the obstacle color is set to gray to highlight the trajectory, instead of changing along the *Z*-axis direction, as in other tests. The start point is set as (0 m, −20 m, 1 m), and several points are randomly selected by 3D Nav Goal tool in Rviz for flight test; the flight process and the trajectory and the speed distribution are as shown in Figure 7. Through the simulation results, it can be seen that the UAV can complete the navigation task in the random scene to perform the flight task normally and completely.

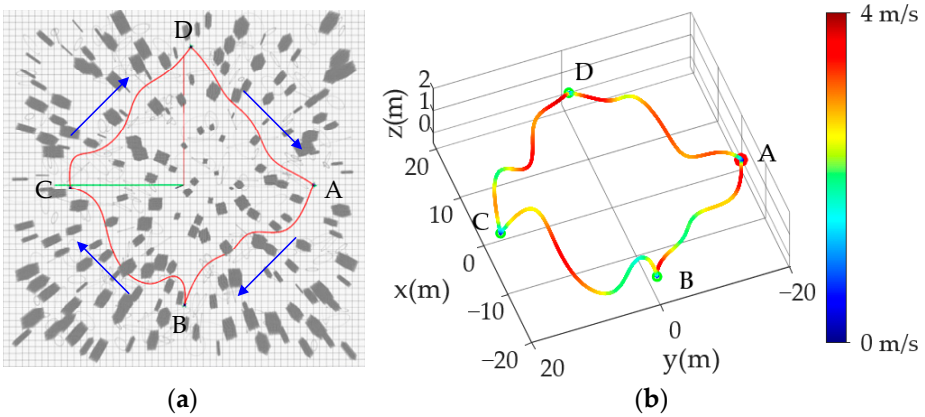


Figure 7. Demonstration of feasibility verification simulation. (**a**) Demonstration plot of randomly selected multiple targets (The red point A is both start and end point, and the green points B–D, selected according to the order of the blue arrows, are randomly selected intermediate target points); (**b**) UAV flight trajectories and the distribution of its velocity.

In order to make the results more general, a test is conducted in an environment with a size of 50 m × 100 m × 5 m, which contains 300 randomly distributed circular and cylindrical obstacles. As shown in Figure 8, the start point of the UAV (red pentagram) is set as (−22 m, 40 m, 1 m), and then the intermediate target points (orange squares) are randomly selected one by one through the 3DGoal tool in the interactive interface of the Rviz visualization, in which the green pentagram is the end point of the whole mission. It can be seen that the UAV does not collide with obstacles throughout the whole process, which indicates that the algorithms in this paper can successfully complete the mission.

Simulation 2: Trajectory comparative analysis. In order to verify the effectiveness of the method in this paper, the trajectories around the vertices of the guided graph of the environment are targeted for comparative analysis. Specifically, the size of the simulation environment is 50 m × 70 m × 5 m, the number of obstacles is 150, and the start point as well as the end point are set as (0 m, −20 m, 1 m) and (0 m, 20 m, 1 m), respectively. For the convenience of description, as shown in Figure 9, the trajectory color of both algorithms is uniformly set to red, the trajectories around the vertices are marked by blue dashed boxes, and the guided graph is indicated by green lines. Since the spherical space for position checking of the state points is delineated around the vertices, it can be seen in Figure 9 that the improved algorithm in this paper generates trajectories further away from the obstacles around the vertices, as compared to [10]. This indicates further that the algorithm proposed in this paper is able to reduce the risk of UAV flight around obstacles in order to ensure the safe flight of UAV.

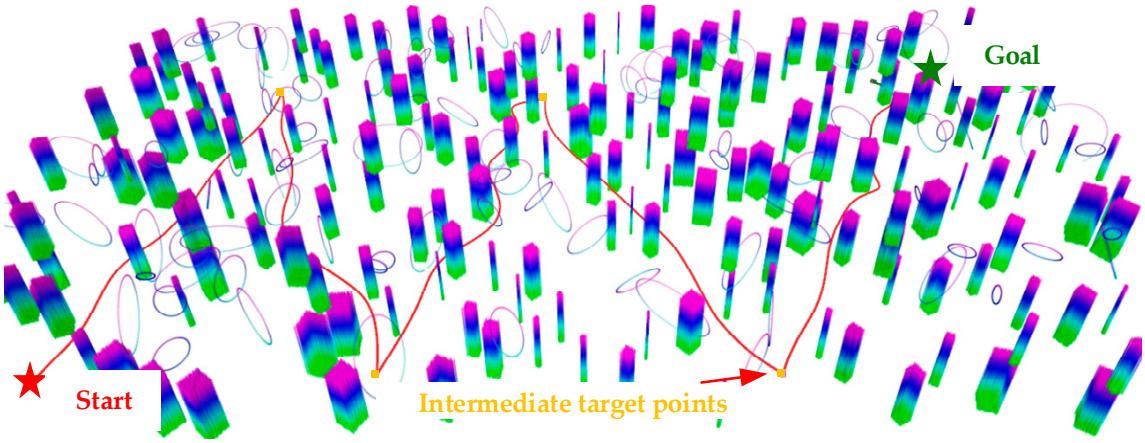
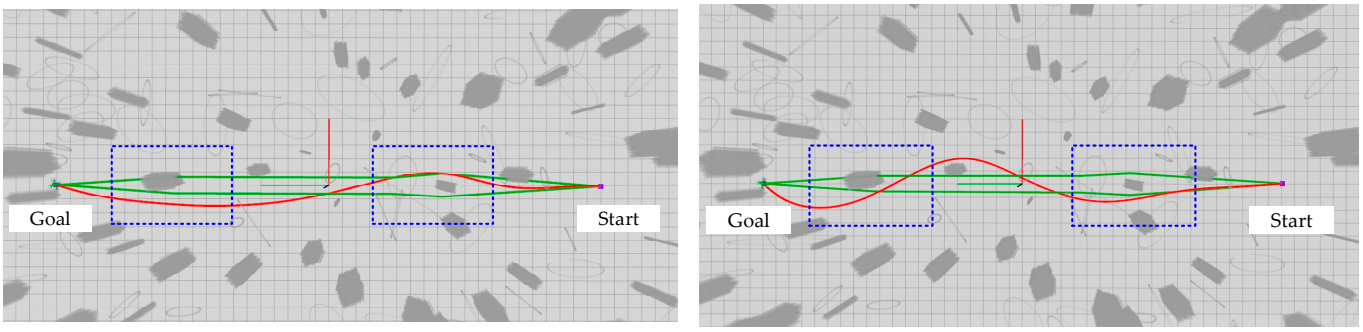


Figure 8. Feasibility validation in dense obstacle environments.



(**a**) (**b**)

Figure 9. Comparative testing of trajectories around vertices of the guided graph of the environment. (**a**) Ref [10]; (**b**) Proposed.

In order to further verify the difference in the effectiveness of trajectories generated by kinodynamic RRT* algorithm with the addition of position checking in this paper, it is compared to the algorithm without the addition [10], in a comparison test conducted in an environment with a size of 60 m × 20 m × 5 m. Start and end points are fixed at (0 m, −20 m, 1 m) and (0 m, 20 m, 1 m), respectively. The number of obstacles is 200 and the resolution of the map is set as 0.1 m. Note from the results shown in Figure 10 that the length of the trajectory of the algorithm proposed in this paper is slightly longer than that of the comparison algorithm. However, as shown by the green dashed box in Figure 10a, the algorithm proposed in this paper clearly bypasses the narrowslit consisting of multiple obstacles. This also shows that, by increasing the position checking of the sampling points, the generated reference trajectory will avoid the more dangerous areas, thus reducing the risk of UAV collisions. In addition, Figure 11 gives the corresponding velocity and acceleration of the above trajectory, and the results further demonstrate the feasibility of the algorithm proposed in this paper.

Simulation 3: Performance analysis. The comparative simulation is validated in a cluttered environment with a map size of 50 m × 50 m × 5 m and a number of obstacles of 200. The start point is set as (0 m, −20 m, 1 m), the end point is set as (−10 m, −15 m, 1 m), and the resolution of the map is set as 0.1 m. Comparing 30 sets of comparative simulation experiments with [10], the computation time as well as the cost of the front-end and back-end trajectories are recorded, and the distributions are visualized in Figure 12a,b.

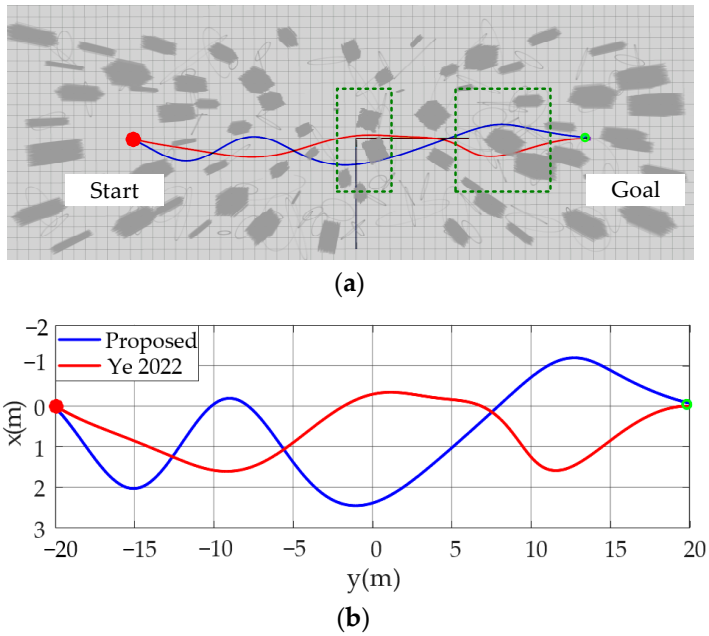


Figure 10. Comparison of the proposed method in this paper with the trajectory of [10]. (**a**) Rviz visualization results for both algorithms; (**b**) UAV trajectory in the X−Y plane.

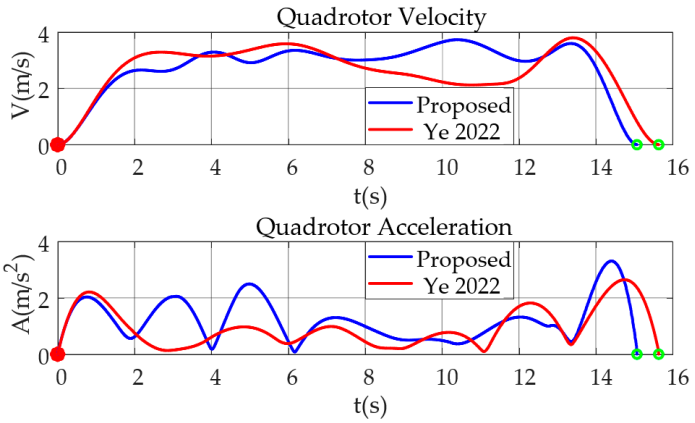


Figure 11. The proposed method in this paper compared with [10] for UAV velocity and acceleration comparison test. (The red dot is the start point and the green dot is the end point.).

During the generation of the front-end trajectory, some points that may make the trajectory too close to the obstacle are discarded by position checking. Therefore, as shown in Figure 12c, the algorithm proposed in the paper has a smaller front-end solution time and computational cost compared with [10]. In addition, it is also shown in Figure 12c that the distribution of the results of the algorithm proposed in this paper is more centralized and the median is also smaller. Similarly, since the local optimization is also adopted in the front-end, the trajectory of the front-end already basically meets the flight requirements, so the trajectory optimization process of the back-end is triggered fewer times. As shown in Figure 12d, the computational cost of the algorithm in the paper is not much different from that of [10], but the computational time is reduced and the median computational time of the algorithm in the paper is lower.

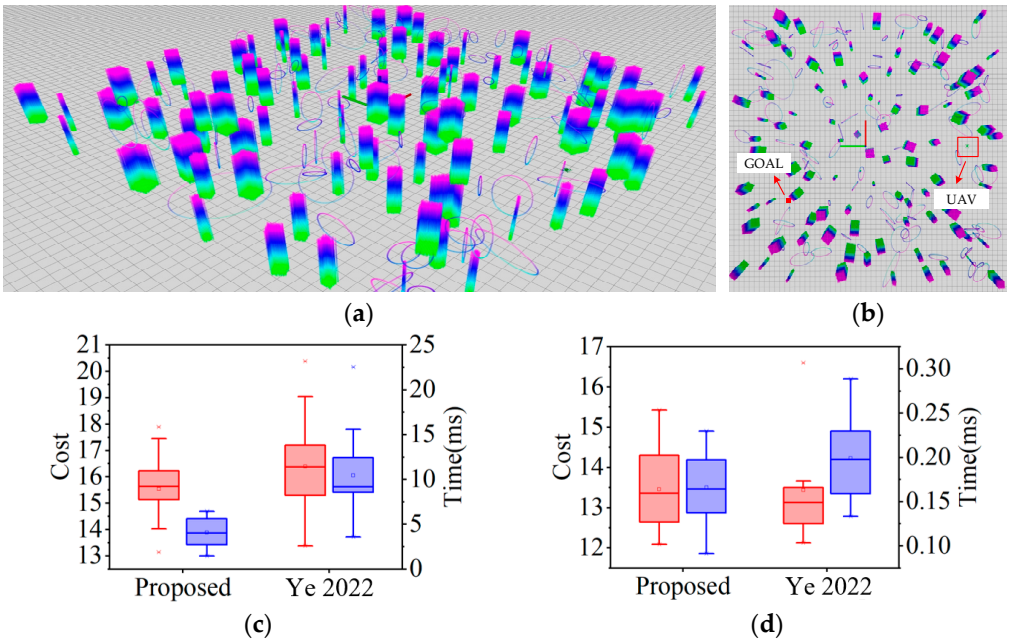


Figure 12. Performance comparison validation results. (**a**,**b**) Simulation environment; (**c**) Comparison of the distribution of front-end computational cost (red) and computational time (blue), with the highlighted horizontal line being the median; (**d**) Comparison of the distribution of back-end computational cost (red) and computational time (blue), with the highlighted horizontal line being the median [10].

5.2. Real-World Experiment

In order to test the effectiveness in executing a real flight mission, the algorithm proposed in this paper is implemented on the vision-based navigation UAV platform. As shown in Figure 13a, the experimental platform is constructed by referring to the Fast-Drone-250 project, with Intel RealSense D435i for perception and mapping in unknown environments. In addition, the onboard computer is Jetson Xavier NX (T503), with 8 GB RAM, 128 G of memory spaceand a pre-installed Ubuntu 18.04 operating system.

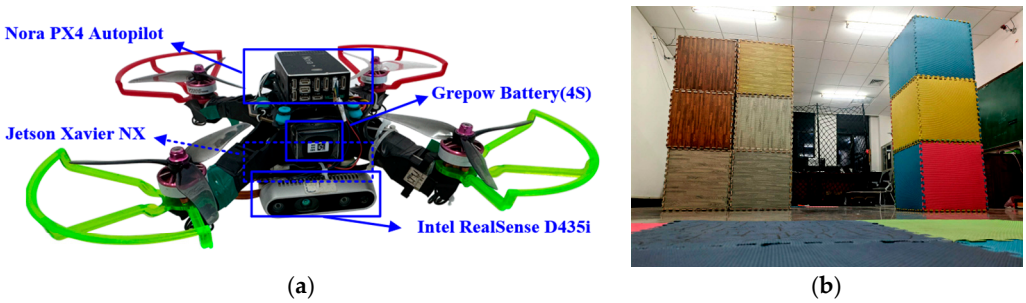


Figure 13. (**a**) Experimental platform; (**b**) Experimental scenario.

Experiment 1: Performance analysis. Experiments are conducted in a specific environment to validate the effectiveness of the algorithm proposed in this paper. Specifically, the end point is set as (4 m, 0 m, 1 m), and the thresholds of velocity and acceleration are set as 1.0 m/s and 0.8 m/s^2, respectively. The work in this paper is based on the framework of [10] and so, in order to verify the effectiveness of the improved algorithm, two algorithms are executed separately, as depicted in the scenarios shown in Figure 13b. Figure 14a shows the grayscale image of the UAV viewpoint at a certain moment, which shows that the UAV is flying relatively stable and without shaking. Figure 14c,d show the captured picture of the third viewpoint at the same moment and the built-up picture in Rviz, respectively. The

specific flight data results are displayed in Figure 14b, from which it can be seen that, in the process of avoiding the obstacles on the right side of the UAV, the algorithm in [10] deviates from the distance length of roughly 0.8~1 m, while the method in this paper deviates from the distance length of 1.2–1.5 m. Therefore, the results demonstrate the potential performance of the algorithm proposed in this paper to ensure the safe flight of UAVs.

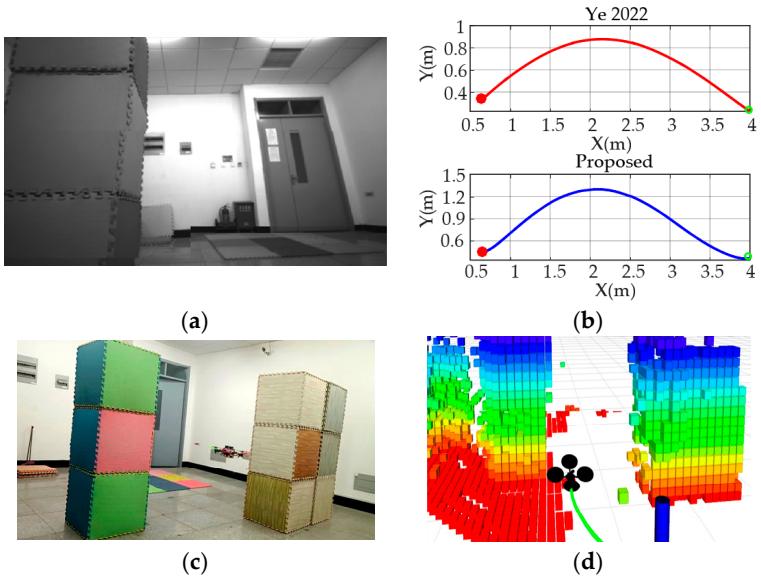


Figure 14. Comparing the test procedure as well as the demonstration results. (**a**) A snapshot of the UAV in flight; (**b**) The trajectories of two algorithms in the same test scenario (red: Ref [10], blue: the method proposed in this paper); (**c**) Image captured by the camera of the UAV in flight; (**d**) The display results in Rviz.

Experiment 2: Universality of the algorithm. In order to verify the effectiveness of the method proposed in this paper in a general scenario, a dense experimental environment with obstacles randomly placed in Figure 15 is constructed. The endpoint is set to (4.5 m, 0 m, 1 m), and the velocity and acceleration thresholds are set to 0.8 m/s and 1.0 m/s^2, respectively. Figure 16a shows the process of the UAV in crossing the dense obstacle environment. Figure 16c,d are the binarized pictures of the UAV viewpoint in the red circle in Figure 16a, as well as the grayscale map, from which it can be seen that the UAV is flying more smoothly without violent shaking. The entire flight trajectory of the UAV and the velocity distribution on the trajectory are shown in Figure 16b, from which it can be seen that the velocity distribution is more uniform. Therefore, it also shows that the algorithm proposed for this problem has the potential to ensure the safety of UAV flight in a cluttered environment.



Figure 15. Randomly placed dense environments. (**a**) Start shooting views; (**b**) End shooting views.

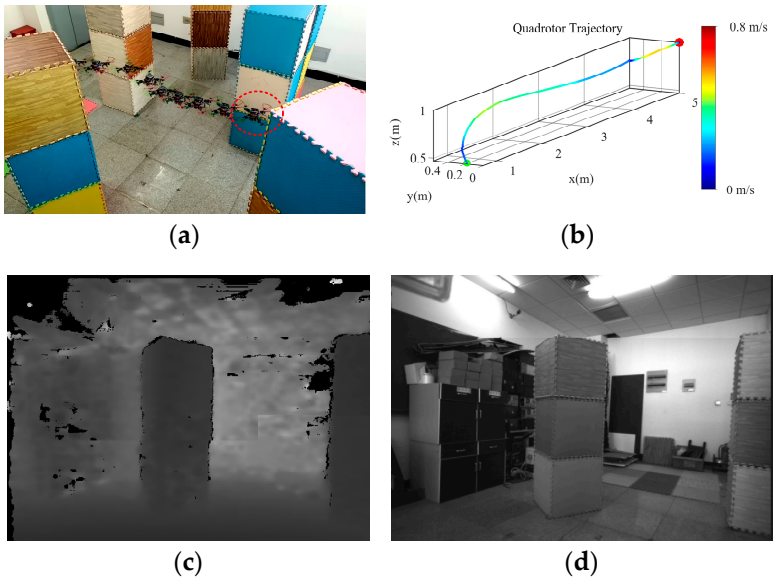


Figure 16. Experimental results in a randomized scenario. (**a**) Diagram of the UAV flight process; (**b**) The corresponding trajectory containing the velocity distribution; (**c**) Depth map of the UAV viewpoint circled with red in (**a**); (**d**) The corresponding grayscale map for the same viewpoint.

6. Conclusions

In this paper, by improving the tree extension and rewriting part of the classical kinodynamic RRT*, a position-checking strategy is proposed to be added to reduce the flight risk of UAVs as well as the unnecessary BVP solving. In addition, the reliability of UAVs in cluttered environments is also improved by local trajectory optimization based on attracting points. Firstly, the traditional RRT* algorithm is analyzed by introducing two core steps, namely, node tree expansion and node rewiring, as well as the existence of the problem that does not consider the body dynamics. Next, on the framework of the classical KRRT* algorithm, a position-checking strategy for new sampling points is proposed, which not only reduces the risk of trajectories around the vertices of the topology-guided graph but also reduces the unnecessary BVP solving. In order to improve the adaptability in cluttered environments, an algorithm for local trajectory optimization is proposed to select different attraction points according to the environment. Finally, it is verified through simulations and experiments that, compared to the existing work, the UAV motion planning algorithm proposed in this paper has a more secure trajectory, and the computational time and computational cost are reduced to some extent. Although the algorithm in this paper combines a local search-based algorithm, it is still a sampling-based algorithm overall and, due to the asymptotic optimality of the sampling algorithm, this leads to the occurrence of failing to find the optimal trajectory within a certain period of time. In the future, in order to improve the optimality of the trajectory, two aspects of research will be considered, namely, the selection of the trigger conditions of the search algorithm and the probability distribution of the sampling algorithm, so that the algorithm in this paper can be more effectively and efficiently applied in complex scenarios, enabling wider applications.

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

Funding: This work is supported by the National Natural Science Foundation of China (grant number 61973234 and 62203326), and in part by the Tianjin Natural Science Foundation (grant number 20JCYBJC00180). The authors would like to acknowledge the anonymous reviewers who offered helpful comments on the manuscript.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The source code presented in this study is available on request from the corresponding author.

Acknowledgments: The authors would like to thank Tiangong University for its technical support, along with all members of our team for their contribution to the UAV flight experiments.

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

References

1. Eldeeb, E.; de Souza Sant, J.M.; Pérez, D.E.; Shehab, M.; Mahmood, N.H.; Alves, H. Multi-UAV Path Learning for Age and Power Optimization in IoT with UAV Battery Recharge. *IEEE Trans. Veh. Technol.* **2023**, *72*, 5356–5360. [CrossRef]
2. Guzzi, J.; Chavez-Garcia, R.O.; Nava, M.; Gambardella, L.M.; Giusti, A. Path Planning with Local Motion Estimations. *IEEE Robot. Autom. Lett.* **2020**, *5*, 2586–2593. [CrossRef]
3. Wen, J.; Zhang, X.; Gao, H.; Yuan, J.; Fang, Y. E^{3}MoP: Efficient Motion Planning Based on Heuristic-Guided Motion Primitives Pruning and Path Optimization with Sparse-Banded Structure. *IEEE Trans. Autom. Sci. Eng.* **2022**, *19*, 2762–2775. [CrossRef]
4. Zhang, S.; Li, Y.; Ye, F.; Geng, X.; Zhou, Z.; Shi, T. A Hybrid Human-in-the-Loop Deep Reinforcement Learning Method for UAV Motion Planning for Long Trajectories with Unpredictable Obstacles. *Drones* **2023**, *7*, 311. [CrossRef]
5. Hsieh, T.-L.; Jhan, Z.-S.; Yeh, N.-J.; Chen, C.-Y.; Chuang, C.-T. An Unmanned Aerial Vehicle Indoor Low-Computation Navigation Method Based on Vision and Deep Learning. *Sensors* **2024**, *24*, 190. [CrossRef] [PubMed]
6. Mostafa, S.; Ramirez-Serrano, A. Three-Dimensional Flight Corridor: An Occupancy Checking Process for Unmanned Aerial Vehicle Motion Planning inside Confined Spaces. *Robotics* **2023**, *12*, 134. [CrossRef]
7. Tordesillas, J.; Lopez, B.T.; Everett, M.; How, J.P. FASTER: Fast and Safe Trajectory Planner for Navigation in Unknown Environments. *IEEE Trans. Robot.* **2022**, *38*, 922–938. [CrossRef]
8. Zhou, B.; Gao, F.; Wang, L.; Liu, C.; Shen, S. Robust and Efficient Quadrotor Trajectory Generation for Fast Autonomous Flight. *IEEE Robot. Autom. Lett.* **2019**, *4*, 3529–3536. [CrossRef]
9. Ye, H.; Zhou, X.; Wang, Z.; Xu, C.; Chu, J.; Gao, F. TGK-Planner: An Efficient Topology Guided Kinodynamic Planner for Autonomous Quadrotors. *IEEE Robot. Autom. Lett.* **2021**, *6*, 494–501. [CrossRef]
10. Ye, H.; Pan, N.; Wang, Q.; Xu, C.; Gao, F. Efficient Sampling-based Multirotors Kinodynamic Planning with Fast Regional Optimization and Post Refining. In Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Kyoto, Japan, 23–27 October 2022; pp. 3356–3363. [CrossRef]
11. Aggarwal, S.; Kumar, N. Path Planning Techniques for Unmanned Aerial Vehicles: A Review, Solutions, and Challenges. *Comput. Commun.* **2020**, *149*, 270–299. [CrossRef]
12. Quan, L.; Han, L.; Zhou, B.; Shen, S.; Gao, F. Survey of UAV motion planning. *IET Cyber Syst Robot.* **2020**, *2*, 14–21. [CrossRef]
13. Tang, G.; Liu, P.; Hou, Z.; Claramunt, C.; Zhou, P. Motion Planning of UAV for Port Inspection Based on Extended RRT* Algorithm. *J. Mar. Sci. Eng.* **2023**, *11*, 702. [CrossRef]
14. Jang, K.; Baek, J.; Park, S.; Park, J. Motion Planning for Closed-Chain Constraints Based on Probabilistic Roadmap with Improved Connectivity. *IEEE-ASME Trans. Mechatron.* **2022**, *27*, 2035–2043. [CrossRef]
15. Lathrop, P.; Boardman, B.; Martínez, S. Distributionally Safe Path Planning: Wasserstein Safe RRT. *IEEE Robot. Autom. Lett.* **2022**, *7*, 430–437. [CrossRef]
16. Gammell, J.D.; Barfoot, T.D.; Srinivasa, S.S. Informed Sampling for Asymptotically Optimal Path Planning. *IEEE Trans. Robot.* **2018**, *34*, 966–984. [CrossRef]
17. Arslan, O.; Tsiotras, P. Use of Relaxation Methods in Sampling-based Algorithms for Optimal Motion Planning. In Proceedings of the IEEE International Conference on Robotics and Automation (ICRA), Karlsruhe, Germany, 6–10 May 2013; pp. 2421–2428. [CrossRef]
18. Boeuf, A.; Cortés, J.; Alami, R.; Siméon, T. Enhancing Sampling-based Kinodynamic Motion Planning for Quadrotors. In Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Hamburg, Germany, 28 September–2 October 2015; pp. 2447–2452. [CrossRef]
19. Karaman, S.; Walter, M.R.; Perez, A.; Frazzoli, E.; Teller, S. Anytime Motion Planning using the RRT*. In Proceedings of the IEEE International Conference on Robotics and Automation (ICRA), Shanghai, China, 9–13 May 2011; pp. 1478–1483. [CrossRef]
20. Gao, F.; Lin, Y.; Shen, S. Gradient-based Online Safe Trajectory Generation for Quadrotor Flight in Complex Environments. In Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Vancouver, BC, Canada, 24–28 September 2017; pp. 3681–3688. [CrossRef]

21. Wang, J.; Chi, W.; Li, C.; Wang, C.; Meng, M.Q.H. Neural RRT*: Learning-Based Optimal Path Planning. *IEEE Trans. Autom. Sci. Eng.* **2020**, *17*, 1748–1758. [CrossRef]
22. Wang, J.; Jia, X.; Zhang, T.; Ma, N.; Meng, M.Q.H. Deep Neural Network Enhanced Sampling-Based Path Planning in 3D Space. *IEEE Trans. Autom. Sci. Eng.* **2022**, *19*, 3434–3443. [CrossRef]
23. Liu, S.; Atanasov, N.; Mohta, K.; Kumar, V. Search-based Motion Planning for Quadrotors using Linear Quadratic Minimum Time Control. In Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Vancouver, BC, Canada, 24–28 September 2017; pp. 2872–2879. [CrossRef]
24. Mellinger, D.; Kumar, V. Minimum Snap Trajectory Generation and Control for Quadrotors. In Proceedings of the IEEE International Conference on Robotics and Automation (ICRA), Shanghai, China, 9–13 May 2011; pp. 2520–2525. [CrossRef]
25. Richter, C.; Bry, A.; Roy, N. Polynomial Trajectory Planning for Aggressive Quadrotor Flight in Dense Indoor Environments. In Proceedings of the The International Symposium of Robotics Research (ISRR), Singapore, 16–19 December 2013; pp. 649–666. [CrossRef]
26. Gao, F.; Wang, L.; Zhou, B.; Zhou, X.; Pan, J.; Shen, S. Teach-Repeat-Replan: A Complete and Robust System for Aggressive Flight in Complex Environments. *IEEE Trans. Robot.* **2020**, *36*, 1526–1545. [CrossRef]
27. Zhou, B.; Gao, F.; Pan, J.; Shen, S. Robust Real-time UAV Replanning Using Guided Gradient-based Optimization and Topological Paths. In Proceedings of the IEEE International Conference on Robotics and Automation (ICRA), Paris, France, 31 May–31 August 2020; pp. 1208–1214. [CrossRef]
28. Zhou, B.; Pan, J.; Gao, F.; Shen, S. RAPTOR: Robust and Perception-Aware Trajectory Replanning for Quadrotor Fast Flight. *IEEE Trans. Robot.* **2021**, *37*, 1992–2009. [CrossRef]
29. Zhou, X.; Wang, Z.; Ye, H.; Xu, C.; Gao, F. EGO-Planner: An ESDF-Free Gradient-Based Local Planner for Quadrotors. *IEEE Robot. Autom. Lett.* **2021**, *6*, 478–485. [CrossRef]
30. Webb, D.J.; Berg, J.v.d. Kinodynamic RRT*: Asymptotically Optimal Motion Planning for Robots with Linear Dynamics. In Proceedings of the IEEE International Conference on Robotics and Automation (ICRA), Karlsruhe, Germany, 6–10 May 2013; pp. 5054–5061. [CrossRef]
31. Ding, W.; Gao, W.; Wang, K.; Shen, S. Trajectory Replanning for Quadrotors Using Kinodynamic Search and Elastic Optimization. In Proceedings of the IEEE International Conference on Robotics and Automation (ICRA), Brisbane, Australia, 21–25 May 2018; pp. 7595–7602. [CrossRef]
32. Gao, F.; Wu, W.; Pan, J.; Zhou, B.; Shen, S. Optimal Time Allocation for Quadrotor Trajectory Generation. In Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Madrid, Spain, 1–5 October 2018; pp. 4715–4722. [CrossRef]

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.