Hindawi
Mathematical Problems in Engineering
Volume 2019, Article ID 5157403, 12 pages
https://doi.org/10.1155/2019/5157403

Research Article

Biased Sampling Potentially Guided Intelligent Bidirectional RRT* Algorithm for UAV Path Planning in 3D Environment

Xiaojing Wu,[1,2] **Lei Xu**,[1] **Ran Zhen**,[1,2] **and Xueli Wu**[1,2]

[1]*School of Electrical Engineering, Hebei University of Science and Technology, Shijiazhuang, Hebei 050018, China*
[2]*Hebei Engineering Technology Research Center of Production Process Automation, Shijiazhuang, Hebei 050018, China*

Correspondence should be addressed to Xueli Wu; xlwu0311@163.com

Received 10 July 2019; Revised 16 September 2019; Accepted 30 September 2019; Published 12 November 2019

Academic Editor: Javier Moreno-Valenzuela

During the last decade, Rapidly-exploring Random Tree star (RRT*) algorithm based on sampling has been widely used in the field of unmanned aerial vehicle (UAV) path planning for its probabilistically complete and asymptotically optimal characteristics. However, the convergence rate of RRT* as well as B-RRT* and IB-RRT* is slow for these algorithms perform pure exploration. To overcome the weaknesses above, Biased Sampling Potentially Guided Intelligent Bidirectional RRT* (BPIB-RRT*) algorithm is proposed in this paper, which combines the bidirectional artificial potential field method with the idea of bidirectional biased sampling. The proposed algorithm flexibly adjusts the sampling space, greatly reduces the invalid spatial sampling, and improves the convergence rate. Moreover, the deeply theoretical analysis of the proposed BPIB-RRT* algorithm is given regarding its probabilistic completeness, asymptotic optimality, and computational complexity. Finally, compared to the latest UAV path planning algorithms, simulation comparisons are demonstrated to show the superiority of our proposed BPIB-RRT* algorithm.

1. Introduction

Nowadays, Unmanned Aerial Vehicles (UAV) are being widely applied to military or civilian fields, e.g., mining surveillance [1], search and rescue operations [2], image recognition [3], cropland assessment [4], and so on. The increasing demands have brought in focus on reducing dependencies on human operators. Path planning plays an important role in enhancing the ability of autonomous and intelligent flight of UAV. In general, given the start and target locations, the goal of path planning is defined as planning a feasible path while satisfying certain criteria, such as distance, smoothness, and obstacle avoidance. Various methods for solving the path planning problem have been raised, such as Rapidly-exploring Random Tree (RRT) [5], Rapidly-exploring Random Tree star (RRT*) [6], ant colony optimization [7], particle swarm optimization [8], simulated annealing method [9], and neural network method [10].

In recent years, RRT and RRT* algorithms have been widely applied to solve the problem of path planning, such as [11, 12], as they performed very well without the information of obstacle configuration in high-dimensional spaces. Different from the traditional RRT algorithm, RRT* algorithm obtains the local optimal path by updating the old parent node constantly, which makes the global path tend to be optimal. Moreover, RRT* can provide a probabilistically complete and asymptotically optimal path. However, RRT and RRT* algorithms perform pure exploration, which can cause it to have very slow rate of convergence in highly dimensional environment. Moreover, the randomly sampled characteristics of RRT and RRT* algorithms make the generated path not smooth enough.

In order to improve the convergence rate, variant improvements on RRT and RRT* algorithms have been extensively explored [13–15]. In [13], the D* Lite Rapidly-exploring Random Tree star (DL-RRT*) algorithm has been proposed for path replanning in dynamic radioactive environments, which uses the expansion strength of grid search strategy from D* Lite to quickly find a high-quality initial path to accelerate the convergence rate in RRT*. However, the path planning method used in [13] was based on single-tree version, so that there is still limitation in the

convergence rate. In the work [14], the authors proposed the Bidirectional RRT (B-RRT) to build two trees rooted at the start and goal configurations, respectively. Reference [15] has extended B-RRT to exploit the possibilities of dynamically dividing up the whole map into four submaps and growing the random tree in every submap using Graphic Processing Unit (GPU) threads. In the other words, Quad-RRT [15] computed four random trees instead of two trees built by bidirectional approaches. The bidirectional tree version of RRT* known as Bidirectional RRT* (B-RRT*) has been presented in [16], which employed a greedy connect heuristic for the connection of two directional trees. However, above-mentioned multiple tree versions of RRT and RRT* performed pure space exploration without using biased sample to guide random trees. Meanwhile, aforementioned algorithms are mainly effective for path planning in 2D environment.

It is well known that a feasible path of UAV should be planned in 3D environment, for 3D environment is more applicable to low-altitude and terrain-following flight. The 3D environment path planning problem has been widely investigated in [17–20]. A bidirectional spline-RRT* path planning algorithm has been proposed in [17] for fixed-wing UAVs that can find paths in highly constrained planning environments. The proposed algorithm can improve the smoothness of the generated path, but does nothing to accelerate the convergence rate. The Intelligent Bi-directional-RRT* (IB-RRT*) algorithm has been proposed in [18] for complex cluttered environments, which introduces an intelligent sample insertion heuristic for fast convergence to the optimal path solution by using uniform sampling heuristics. However, aforementioned variants of the RRT* algorithm still perform pure exploration without guide for sampling, which should make the path planning algorithm to have slow convergence rate and suffer in highly cluttered environments. Reference [19] presented the Potential Function Based-RRT* (P-RRT*) algorithm for better convergence properties by introducing a Random Gradient Descent (RGD) heuristic. The RGD heuristic guides the randomly sampled states incrementally downhill in the direction of decreasing potential so that the convergence rate is improved. Then, the new bidirectional potential gradient heuristics was presented in [20] to potentially guide two rapidly-exploring random trees towards each other in the bidirectional sampling-based motion planning, and hence the two algorithms called Potentially Guided Bidirectional RRT* (PB-RRT*) and Potentially Guided Intelligent Bi-directional RRT* (PIB-RRT*) greatly improved the sampling efficiency, resulting in faster convergence rate and excellent performance in highly cluttered environment. However, the biased sampling technique was not introduced in the PIB-RRT* and PB-RRT* algorithms [20]. Moreover, the fixed attractive pole may give misleading information that points the random trees in the wrong direction. By introducing the idea of target gravity, the tGSRT algorithm was presented in [5]. The speed of the initial path search was increased, and the path was optimized by the genetic algorithm (GA) and smooth processing. Nevertheless, there are some limitations in the sampling procedure, and the path generated with jags is inevitable.

In this paper, we present the concept of Biased Sampling Potentially Guided Intelligent Bidirectional RRT* (BPIB-RRT*) for path planning of UAV, guiding two random trees towards each other by incorporating the proposed bi-directional biased sampling and bidirectional potential field heuristics. This paper presents the bidirectional biased sampling heuristic to provide a higher deviation for sampling in the right direction based on the relative position relation between the target trees with the sampling. Also, dynamic attractive pole based on the growth trend of the target trees as the bidirectional potential field heuristic is introduced into BI-RRT*, which pulls the random trees towards each other fast. The idea of the BPIB-RRT* algorithm reduces the invalid nodes generated by random sampling in state space and enhances the guidance, which can improve the sampling efficiency of the algorithm. Moreover, the generated path is smoothed by B-spline to meet the path constraint conditions of UAV, which improves the flight ability. The algorithm is novel and can greatly improve the convergence rate and smoothness of the planning path for 3D environment.

The remainder of the paper is divided into the following sections: Section 2 provides the explanation of symbols which are used to describe the problem. Section 3 gives the related works and backgrounds of the BPIB-RRT* algorithm. Section 4 explains the BPIB-RRT* algorithm proposed in this paper. Section 5 presents the theoretical analyses of the BPIB-RRT* algorithm. Section 6 gives simulation results regarding the comparison of data and figures. Finally, conclusions are drawn in Section 7.

2. Problem Definition

This section will describe the notations that are used in this paper. The state space is represented by the set $S \subset R^3$. Furthermore, the given configuration space is divided into obstacle regions and obstacle-free regions which are denoted by $S_{\text{obs}} \subset S$ and $S_{\text{free}} = S \backslash S_{\text{obs}}$, respectively. Let $T_a = (V_a, E_a) \subset S_{\text{free}}$ and $T_b = (V_b, E_b) \subset S_{\text{free}}$ represent two random trees which are initialized by $S_{\text{init}} \in S_{\text{free}}$ and $S_{\text{goal}} \in S_{\text{free}}$ as their root vertex, respectively. V_a, V_b, E_a, and E_b denote the vertices and edges of two trees T_a and T_b. The closed ball of radius $r > 0$ centred at s' is defined as $O_{s',r} := \{s \in S \mid \|s - s'\| \leq r\}$. Let k denote the number of iterations. Let $\{\eta_N\}$ be the set of feasible path produced by an algorithm where $\{\eta_N\} \in S_{\text{free}}$ and N is the number of a complete path vertexes. The problem of UAV path planning is to find a feasible path $\{\eta_N\}$ from the initial state to the target location in the least amount of time possible in the 3D environment. A complete feasible path is denoted by $\eta[0,1] \longrightarrow \{\eta(0) = S_{\text{init}}, \eta(1) = S_{\text{goal}}\} \subset S_{\text{free}}$.

Let the cost function J denote the cumulative sum of cost (in terms of Euclidean distance) of coordinate (x_n, y_n, z_n) to coordinate $(x_{n+1}, y_{n+1}, z_{n+1})$, and this calculation process continues till $n \longrightarrow N$. The cost function J can be defined as

$$J = \sum_{n=1}^{N} L(n), \tag{1}$$

where

$$L(n) = \left((x_{n+1} - x_n)^2 + (y_{n+1} - y_n)^2 + (z_{n+1} - z_n)^2\right)^{1/2}. \tag{2}$$

3. Related Works

In this section, various RRT* algorithms are described briefly.

3.1. RRT Algorithm.* The RRT* [6] algorithm is explained in detail in Algorithm 1. The RRT* algorithm initializes the random tree $T = (V, E)$ with initial location S_{init} to start the iterative process (Line 1). The random sample $q_{rand} \in S$ is given by *Sample* () (Line 3). Then, the near vertices or the nearest vertex of q_{rand} is obtained by *NearVertices*() or *NearestVertex*() (Line 4–7). The populated set Q_{near} is then sorted by the *GetSortedList*() procedure (Line 8). *ChooseBestParent*() returns $q_{\min} \in Q_{\text{near}}$ with minimum cost from the random tree T to q_{rand} in obstacle-free space (Line 9). Once the best parent vertex is located, $q_{\min}$ is inserted into the random tree $T = (V, E)$ by making q_{rand} its child and the rewiring procedure is executed (Line 10–13). Algorithm iteration continues till $k \longrightarrow K$ or the goal location is explored.

3.2. B-RRT Algorithm.* Algorithm 2 is given in this section, which outlines the implementation of Bidirectional RRT* (B-RRT*) in detail.

The B-RRT* [14] algorithm initializes two random trees T_a and T_b in the same manner as they were in the RRT* (Line 1). The B-RRT* [14] algorithm starts with sampling from the obstacle-free space S_{free}, and the same process is acted on this random node q_{rand} (Line 4–9). Then, q_{rand} is inserted into the random tree T_a by *InsertVertex*() (Line 10–13), and the nearest node q_{conn} from tree T_b is found (Line 14). If *Connect*() heuristic successfully connects random trees T_a and T_b, a feasible path η_{new} from S_{init} to S_{goal} is generated (Line 15). The cost of η_{new} is compared with the previously computed path cost η_{best} to confirm an optimal path with less cost (Line 16–18). Then, random trees T_a and T_b are swapped, and the procedures mentioned above are executed on the other tree T_b again.

3.3. IB-RRT Algorithm.* This section explains the implementation of Intelligent Bidirectional RRT* [18] (IB-RRT*). The IB-RRT* algorithm is proposed, which can use the *GetBestTreeParent*() heuristic to obtain the best parent from two trees T_a or T_b.

The IB-RRT* [18] algorithm initializes the random tree T_a and T_b as well as B-RRT*. The random vertex q_{rand} is produced by *Sample*(). Then, the near neighbour vertices $Q^{\text{a}}_{\text{near}}$ and $Q^{\text{b}}_{\text{near}}$ are generated by *NeghboringVertices*() from T_a and T_b, respectively. Both neighbouring vertex sets $Q^{\text{a}}_{\text{near}}$ and $Q^{\text{b}}_{\text{near}}$ are sorted by the *GetSortedList*() procedure. The *GetBestTreeParent*() heuristic returns the best parent $P_{\min}$ from T_a or T_b. Then, the q_{rand} is inserted in the best selected tree, and the rewiring procedure is executed.

3.4. PIB-RRT Algorithm.* The PIB-RRT* [20] algorithm incorporates the Artificial Potential Fields (APF) into IB-RRT* by using the *BPG*() heuristic. The *BPG*() heuristic returns the potentially guided bidirectional randomly sampled state q_{pb}. The *BPG*() heuristic is shown in Algorithm 3 and explained below.

BPG(): if the iteration number k is even, the potentially guided bidirectional randomly sampled state q_{pb} is fed into the *BPGgoal*() heuristic, which is used to calculate the attractive force F_{att} working on q_{pb}. Comparing d^*_{nearest} with a constant d^*_{obs}, q_{pb} will be guided downhill in the direction of decreasing potential towards the goal region. Similarly, for odd iteration k, q_{pb} is passed to the *BPGinit*() heuristic and attractive force F_{att} is calculated. Then, q_{pb} is pulled towards S_{init} by the *BPGinit*() heuristic.

4. Biased Sampling Potentially Guided Intelligent Bidirectional RRT*

In this section, we present our proposed algorithm, Biased Sampling Potentially Guided Intelligent Bidirectional RRT* (BPIB-RRT*).

4.1. Algorithm Process. The detailed procedure of the proposed algorithm is explained as mentioned below.

From aforementioned descriptions, the flow chart of the BPIB-RRT* algorithm can be presented in Figure 1.

Extra procedures employed by BPIB-RRT* are explained below, whereas the rest are exactly the same as they are for IB-RRT*. Therefore, only the *Bi-bias*() heuristic and *BPF*() heuristic are explained in detail.

4.2. Bi-bias(). The random sampling process of the RRT* algorithm based on full space causes its convergence to the optimal solution very slow. Hence, the *Bi-bias*() heuristic based on the growth trend of the target tree is introduced into IB-RRT*.q_{rand} is generated by *Sample*() firstly and then passed to *Bi-bias*(). If the iteration count k is even, then the randomly sampled state q_{rand} is pulled toward the random tree T_b. The bias direction can be obtained by analysing the relative position between q_{prev2} and q_{rand}, where q_{prev2} is the new vertex of the target tree T_b. *The Bi-bias*() heuristic can provide a higher deviation in the right direction before generating the bias node q_{bia}. If the iteration count k is odd, the random tree T_a attracts the randomly sampled state q_{rand}. Similarly, the relative position between q_{prev1} and q_{rand} is calculated to provide a higher deviation for q_{rand}, where q_{prev1} is the new vertex of the target tree T_a.

The bias formulas are shown below:

(1) $V \longleftarrow \{S_{\text{init}}\}$; $E \longleftarrow \emptyset$; $T \longleftarrow (V, E)$;
(2) **for** $k \longleftarrow 0$ **to** K **do**
(3) $q_{rand} \longleftarrow$ Sample(k)
(4) $Q_{\text{near}} \longleftarrow$ *NearVertices*(q_{rand}, T)
(5) **if** $Q_{\text{near}} = \emptyset$ **then**
(6) $Q_{\text{near}} \longleftarrow$ *NearestVertex*(q_{rand}, T)
(7) **end**
(8) $L_s \longleftarrow$ *GetSortedList*(q_{rand}, Q_{near})
(9) $q_{\min} \longleftarrow$ *ChooseBestParent*(L_s)
(10) **if** $q_{\min} \neq \emptyset$ **then**
(11) $T \longleftarrow$ *InsertVertex*(q_{rand}, $q_{\min}$, T)
(12) $T \longleftarrow$ RewireVertices(q_{rand}, L_s, E)
(13) **end**
(14) **end**
(15) **return** $T = (V, E)$

Algorithm 1: RRT* (S_{init}).

(1) $V \longleftarrow \{S_{\text{init}}, S_{\text{goal}}\}$; $E \longleftarrow \emptyset$; $T_a \longleftarrow (S_{\text{init}}, E)$; $T_b \longleftarrow (S_{\text{goal}}, E)$;
(2) $\eta_{\text{best}} \longleftarrow \infty$
(3) **for** $k \longleftarrow 0$ **to** K **do**
(4) $q_{rand} \longleftarrow$ Sample(k)
(5) $q_{\text{nearest}} \longleftarrow$ *NearestVertex*(q_{rand}, T_a)
(6) $q_{\text{new}} \longleftarrow$ Extend(q_{nearest}, q_{rand})
(7) $Q_{\text{near}} \longleftarrow$ *NearVertices*(q_{new}, T_a)
(8) $L_s \longleftarrow$ *GetSortedList*(q_{new}, Q_{near})
(9) $q_{\min} \longleftarrow$ *ChooseBestParent*(L_s)
(10) **if** $q_{\min} \neq \emptyset$ **then**
(11) $T \longleftarrow$ *InsertVertex*(q_{new}, $q_{\min}$, T_a)
(12) $T \longleftarrow$ RewriteVertices(q_{new}, L_s, E)
(13) **end**
(14) $q_{\text{conn}} \longleftarrow$ *NearestVertex*(q_{new}, T_b)
(15) $\eta_{\text{new}} \longleftarrow$ *Connect*(q_{new}, q_{conn}, T_b)
(16) **if** $\eta_{\text{new}} \neq \emptyset$ &&$C(\eta_{\text{new}}) < C(\eta_{\text{best}})$ **then**
(17) $\eta_{\text{best}} \longleftarrow \eta_{\text{new}}$
(18) **end**
(19) SwapTrees(T_a, T_b)
(20) **end**
(21) **return** T_a, $T_b = (V, E)$

Algorithm 2: B-RRT* (S_{init}, S_{goal}).

$$\begin{cases} x_{\text{bia}} = x_{rand} + rand^* \left(x_{\text{prev}*} - x_{rand}\right), & x_{rand} \le x_{\text{prev}*}, \\ x_{\text{bia}} = x_{rand} - rand^* \, x_{rand}, & x_{rand} > x_{\text{prev}*}, \end{cases} \tag{3}$$

$$\begin{cases} y_{\text{bia}} = y_{rand} + rand^* \left(y_{\text{prev}*} - y_{rand}\right), & y_{rand} \le y_{\text{prev}*}, \\ y_{\text{bia}} = y_{rand} - rand^* \, y_{rand}, & y_{rand} > y_{\text{prev}*}, \end{cases} \tag{4}$$

$$\begin{cases} z_{\text{bia}} = z_{rand} + rand^* \left(z_{\text{prev}*} - z_{rand}\right), & z_{rand} \le z_{\text{prev}*}, \\ z_{\text{bia}} = z_{rand} - rand^* \, z_{rand}, & z_{rand} > z_{\text{prev}*}, \end{cases} \tag{5}$$

where $(x_{rand}, y_{rand}, z_{rand})$ denotes the random state coordinate, $(x_{\text{prev1}}, y_{\text{prev1}}, z_{\text{prev1}})$ and $(x_{\text{prev2}}, y_{\text{prev2}}, z_{\text{prev2}})$ represent the new vertex of target trees T_a and T_b, respectively. *rand* denotes the random function between 0 and 1. Bias node $q_{\text{bia}} = (x_{\text{bia}}, y_{\text{bia}}, z_{\text{bia}})$ is calculated by the aforementioned formulas.

(1) $q_{\text{pb}} \longleftarrow q_{rand}$
(2) **if** k mod $2 = 0$ **then**
(3) **for** $k \longleftarrow 0$ **to** n **do**
(4) $F_{\text{att}} =$ BPGgoal (S_{goal}, q_{pb})
(5) $d^*_{\text{nearest}} \longleftarrow$ NearestObstacleSearch(S_{obs}, q_{pb})
(6) **if** $d^*_{\text{nearest}} \le d^*_{\text{obs}}$ **then**
(7) **return** q_{pb}
(8) **else**
(9) $q_{\text{pb}} \longleftarrow q_{\text{pb}} + \varepsilon \ (F_{\text{att}}/\|F_{\text{att}}\|)$
(10) **else**
(11) **for** $k \longleftarrow 0$ **to** n **do**
(12) $F_{\text{att}} = \text{BPG}_{\text{initt}}(S_{\text{init}}, q_{\text{pb}})$
(13) $d^*_{\text{nearest}} \longleftarrow$ NearestObstacleSearch(S_{obs}, q_{pb})
(14) **if** $d^*_{\text{nearest}} \le d^*_{\text{obs}}$ **then**
(15) **return** q_{pb}
(16) **else**
(17) $q_{\text{pb}} \longleftarrow q_{\text{pb}} + \varepsilon \ (F_{\text{att}}/\|F_{\text{att}}\|)$
(18) **return** q_{pb}

Algorithm 3: *BPG*(q_{rand}, k).

It can be seen from Figure 2, there are eight relative positions between $q_{\text{prev}*}$ and q_{rand} in state space. Take the spatial position shown in Figure 2 as an example, part of the *Bi-bias*() heuristic is shown in Algorithm 5. The calculation of other spatial relationships for q_{rand} and $q_{\text{prev}*}$ is the same. By feeding back the vertex position of the two random trees, *Bi-bias*() updates the spatial position of random state q_{rand} dynamically, improves the effectiveness of the sampling process, and furthermore reduces the number of iterations of the algorithm.

4.3. BPF(). Artificial potential field (APF) has a good application in obstacle avoidance technology of UAV. The APF algorithm constructs attractive field and repulsive force field based on the target point and obstacles in the planning space, respectively. The goal region pulls the robot towards it, and the obstacles repel the robot away from the obstacle configuration space S_{obs}. The force exerted to the robot is equal to the resultant force of attractive force and repulsive force at this location [21]. APF is simple to operate and has a fast convergence speed. It could get stuck in the local minima for it performs pure exploitation, and it is greedy. APF is smooth in mathematic which means the tangent and the velocity are always continuous [22] and improves UAV stability. The equation of attraction field is discussed below:

$$U_{\text{Att}}(q) = \frac{1}{2} k_{\text{Att}} \rho^2 \left(q, S_{\text{goal}}\right), \tag{6}$$

where $U_{\text{Att}}(q)$ is the attraction field, k_{Att} is the attraction field coefficient, q is the current position of the UAV, S_{goal} is the goal location, and $\rho(q, S_{\text{goal}})$ is the distance between the current point and the goal location. The attraction field force $F_{\text{Att}}(q)$ is generated by taking a negative gradient

(1) $V_a \longleftarrow \{S_{\text{init}}\}$; $V_b \longleftarrow \{S_{\text{goal}}\}$; $E_a \longleftarrow \emptyset$; $E_b \longleftarrow \emptyset$;
(2) $T_a \longleftarrow (V_a, E_a)$; $T_b \longleftarrow (V_b, E_b)$;
(3) $q_{\text{prev1}} = S_{\text{init}}$
(4) $q_{\text{prev2}} = S_{\text{goal}}$
(5) $\eta_{\text{best}} \longleftarrow \infty$
(6) Connection $\longleftarrow$ True
(7) **for** $k \longleftarrow 0$ **to** K **do**
(8) $q_{rand} \longleftarrow$ Sample(k)
(9) $q_{\text{bia}} \longleftarrow$ *Bi-bias*(q_{rand}, k, q_{prev1}, q_{prev2})
(10) $q_{\text{op}} \longleftarrow$ *BPF*(q_{bia}, k, q_{prev1}, q_{prev2})
(11) $\{Q^a_{\text{near}}, Q^b_{\text{near}}\} \longleftarrow$ *NeighbouringVertices*(q_{op}, T_a, T_b)
(12) **if** $Q^a_{\text{near}} = \emptyset$ *and* $Q^b_{\text{near}} = \emptyset$ **then**
(13) $\{Q^a_{\text{near}}, Q^b_{\text{near}}\} \longleftarrow$ *NearestVertex*(q_{op}, T_a, T_b)
(14) Connection $\longleftarrow$ False
(15) $L_a \longleftarrow$ *GetsortedList*(q_{op}, Q^a_{near})
(16) $L_b \longleftarrow$ *GetsortedList*(q_{op}, Q^b_{near})
(17) $\{q_{\text{parent}}$, flag, $\eta_{\text{best}}\} \longleftarrow$ GetBestTreeParent(L_a, L_b, Connection)
(18) **if** (flag) **then**
(19) $T_a \longleftarrow$ *InsertVertex*(q_{op}, q_{parent}, T_a)
(20) $T_a \longleftarrow$ RewriteVertices(q_{op}, L_a, E_a)
(21) $q_{\text{prev1}} \longleftarrow q_{\text{op}}$
(22) **else**
(23) $T_b \longleftarrow$ *InsertVertex*(q_{op}, q_{parent}, T_b)
(24) $T_b \longleftarrow$ RewriteVertices(q_{op}, L_b, E_b)
(25) $q_{\text{prev2}} \longleftarrow q_{\text{op}}$
(26) $E \longleftarrow E_a \cup E_b$
(25) $V \longleftarrow V_a \cup V_b$
(26) **return** $\{T_a, T_b\} = (V, E)$
(27) $\eta* \longleftarrow$ B-spline(T_a, T_b)

Step 1: Sampling optimization: randomly sampled state q_{rand} is obtained by the sampling process in the given state space. Then, the biased node q_{bia} based on the growth trend of the target random tree is obtained by using the *Bi-bias()* heuristic. After being guided by the *BPF()* heuristic, randomly sampled state q_{rand} turns into biased potentially guided bidirectional randomly sampled state q_{op} such that $q_{rand} \longrightarrow q_{\text{op}}$, where $q_{\text{op}} \in S_{\text{free}}$. *Bi-bias()* and *BPF()* heuristics have been explained later in this section.
Step 2: Insert vertex: the GetBestTreeParent() heuristic finds the best parent for the state q_{op} obtained in Step 1 from random trees T_a or T_b and *InsertVertex*() insert the q_{op} into the best selected tree. Step 1 and Step 2 continue till $k \longrightarrow K$.
Step 3: Generate the path. As iteration, k, goes from 0 to K, a feasible solution path $\eta[0, 1]$ from initial to goal configuration is found. Then, the final path is smoothed by B-spline. Finally, BPIB-RRT* can provide a smooth feasible path. The pseudocode of BPIB-RRT* is given in Algorithm 4.

Algorithm 4: BPIB-RRT*(S_{init}, S_{goal}).

$$F_{\text{Att}}(q) = -\nabla U_{\text{Att}}(q) = k_{\text{Att}}\rho\left(q, S_{\text{goal}}\right). \tag{7}$$

The repulsive field $U_{\text{Rep}}(q)$ is generated by all obstacles $S_{\text{obs}} \in S$.

$$U_{\text{Rep}}(q) = \sum_{i=1}^{I} U^i_{\text{Rep}}(q), \tag{8}$$

where $U^i_{\text{Rep}}(q)$ is the repulsion field generated by the ith obstacle at q.

$$U^i_{\text{Rep}}(q) = \begin{cases} \dfrac{1}{2}\lambda\left(\dfrac{1}{\rho\left(q, q^i_{\text{obs}}\right)} - \dfrac{1}{\rho^i_0}\right)^2, & \rho\left(q, q^i_{\text{obs}}\right) < \rho^i_0, \\ 0, & \rho\left(q, q^i_{\text{obs}}\right) \geq \rho^i_0, \end{cases} \tag{9}$$

where λ is the repulsive field coefficient, ρ^i_0 is the effective radius of the ith repulsive field, and $\rho(q, q^i_{\text{obs}})$ is the distance between the ith obstacle and the current position q. Correspondingly, let $F^i_{\text{Rep}}(q)$ sign the repulsion force,

$$F^i_{\text{Rep}}(q) = \begin{cases} F^i_{\text{Rep1}}(q), & \rho\left(q, q^i_{\text{obs}}\right) < \rho^i_0, \\ 0, & \rho\left(q, q^i_{\text{obs}}\right) \geq \rho^i_0, \end{cases} \tag{10}$$

where $F^i_{\text{Rep1}}(q) = \lambda\left(\left(1/\rho\left(q, q^i_{\text{obs}}\right)\right) - \left(1/\rho^i_0\right)\right)\left(1/\rho^2\left(q, q^i_{\text{obs}}\right)\right) \nabla\rho\left(q, q^i_{\text{obs}}\right)$.

The resultant force on the UAV at point q is $F(q)$.

$$F(q) = F_{\text{Att}}(q) + \sum_{i=1}^{I} F^i_{\text{Rep}}(q). \tag{11}$$

Incorporating APF into IB-RRT* by using the Bidirectional Potential Field heuristic (*BPF*()), pseudocode of the *BPF*() heuristic is presented in Algorithm 6. The current vertices of two random trees q_{prev1} and q_{prev2} are first fed into the *BPF()* algorithm. When the number of iterations k is even, the new vertex q_{prev2} of the T_b is treated as the target point of the bias point q_{bia}, and the attraction force based on

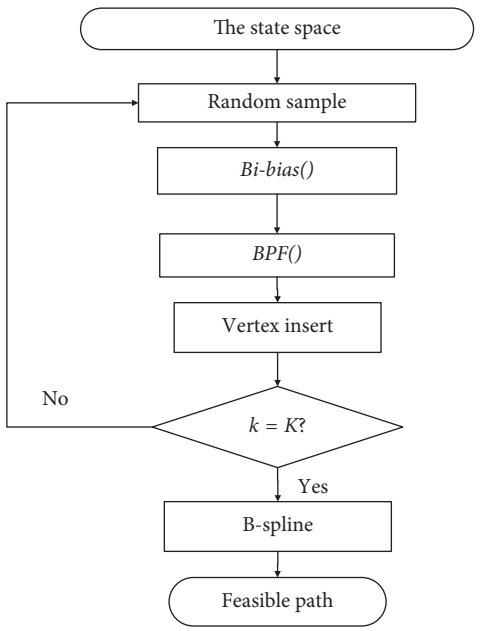

Figure 1: Flow chart of the BPIB-RRT* algorithm.

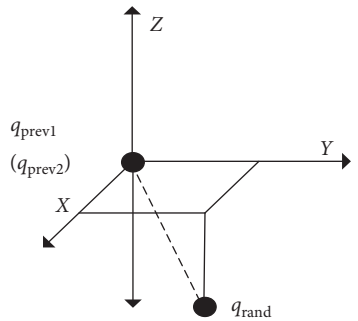

Figure 2: The instance of spatial position.

(1) **if** k **mod** $2 = 0$ **then**
(2) if $(x_{prev2} < x_{rand})$ then
(3) if $(y_{prev2} < y_{rand})$ then
(4) if $(z_{prev2} > z_{rand})$ then
(5) $x_{bia} = x_{rand} - rand^* \, x_{rand}$
(6) $y_{bia} = y_{rand} - rand^* \, y_{rand}$
(7) $z_{bia} = z_{rand} + rand^* \, (z_{prev2} - z_{rand})$
(8) **else**
(9) if $(x_{prev1} < x_{rand})$ then
(10) if $(y_{prev1} < y_{rand})$ then
(11) if $(z_{prev1} > z_{rand})$ then
(12) $x_{bia} = x_{rand} - rand^* \, x_{rand}$
(13) $y_{bia} = x_{rand} - rand^* \, y_{rand}$
(14) $z_{bia} = z_{rand} - rand^* \, (z_{prev1} - z_{rand})$
(15) return $q_{bia} = (x_{bia}, y_{bia}, z_{bia})$
(16) **end**

Algorithm 5: *Bi-bias*(q_{rand}, k, q_{prev1}, q_{prev2}).

the q_{prev2} is calculated. $d^i_{nearest} \in D_{nearest}$ is the distance between the bias point q_{bia} and the ith obstacle. ρ^i_0 denotes the radius of repulsion field. If $d^i_{nearest}$ is less than ρ^i_0, q_{bia} will be guided toward the direction of the resultant force $F(q)$ based on the repulsion force and the attraction force, otherwise only attraction force $F_{Att}(q)$ works on q_{bia}. Then, the optimal random node q_{op} is obtained. If the iteration count k is odd, the *BPF()* heuristic treats the new vertex q_{prev1} of the T_a as the attraction field of the bias point q_{bia}. Similarly, q_{bia} is directed downhill in direction of the decreasing potential by computing the resultant force, and the optimal random node q_{op} is produced.

4.4. Path Generation Using B-Spline. The curvature of path generated by the RRT* algorithm is noncontinuous. Therefore, the initial path cannot meet the flight constraints of UAV. To overcome this shortcoming, B-spline is introduced into BPIB-RRT* due to its high flexibility and arbitrary continuity. The path vertices generated by the BPIB-RRT* algorithm are fed into B-spline as its control points to generate a smooth path with continuous curvature [23]. The equation of the B-spline curve can be presented by the following equation:

$$p(t) = \sum_{i=0}^{n} p_i \bullet F_{i,k}(t), \tag{12}$$

where $p_i \, (i = 0, 1, \ldots, n)$ are the characteristic nodes of the spline curve. The ith B-spline basis function of degree k, written as $F_{i,k} \, (i = 0, 1, \ldots, n)$, can be expressed as follows:

$$\begin{cases} F_{0,3}(t) = \dfrac{1}{6}(1-t)^3, \\ F_{1,3}(t) = \dfrac{1}{6}\left(3t^3 - 6t^2 + 4\right), \\ F_{2,3}(t) = \dfrac{1}{6}\left(-3t^3 + 3t^2 + 3t + 1\right), \\ F_{3,3}(t) = \dfrac{1}{6}t^3, \end{cases} \tag{13}$$

where $t \in [0, 1]$.

5. Analysis

5.1. Probabilistic Completeness. Let $D^a_k(p)$ denote the random variable of the distance with the minimum cost from node $p \in S_{free}$ to the first random tree T_a, and d^a_k is the value of the random variable $D^a_k(p)$, where k is the number of iterations for BPIB-RRT*. Similarly, let $D^b_k(p)$ denote the random variable of the minimum distance cost between nodes $p \in S_{free}$ and T_b , and d^b_k is the value of the random variable. $D_k(p_a, p_b)$ is the random variable of the distance with the minimum distance cost from T_a to T_b, where p_a and p_b belong to T_a and T_b, respectively. $d_k(p_a, p_b)$ is the value of the random variable $D_k(p_a, p_b)$.

Probabilistic Completeness: for a robust feasible path planning problem $(S_{free}, S_{init}, S_{goal})$, when the number of iterations k approaches infinity, the distance between p_a and

```
(1) q_op ⟵ q_bia
(2)   if k mod 2 = 0 then
(3)     F_Att = computeAttForce(q_prev2, q_op)
(4)     D_nearest = ρ(S_obs, q_op)
(5)     if d^i_nearest ≤ ρ^i_0 then
(6)       F_Rep = computeRepForce(q_op, S_obs)
(7)       q_op = q_op + (F_Rep/‖F_Rep‖) + (F_Att/‖F_Att‖)
(8)     else
(9)       q_op = q_op + (F_Att/‖F_Att‖)
(10)    end
(11)  else
(12)    F_Att = computeAttForece(q_prev1, q_op)
(13)    D_nearest = ρ(S_obs, q_op)
(14)    if d^i_nearest ≤ ρ^i_0 then
(15)      F_Rep = computeRepForce(q_op, S_obs)
(16)      q_op = q_op + (F_Rep/‖F_Rep‖) + (F_Att/‖F_Att‖)
(17)    else
(18)      q_op = q_op + (F_Att/‖F_Att‖)
(19)    end
(20) end
(21) return q_op
```

Algorithm 6: BPF (q_{bia}, k, q_{prev1}, q_{prev2}).

p_b is less than a constant τ. The probability of finding a feasible solution goes to one.

$$\lim_{k\longrightarrow\infty} P\left(d_k(p_a, p_b) < \tau \mid p_a, p_b \in S_{\text{free}}\right) = 1. \tag{14}$$

Theorem 1. *The BPIB-RRT* algorithm is a probabilistically complete algorithm. In the three-dimensional solution space S, as the number of iterations approaches infinity, probability of finding a feasible path from the initial to goal region goes to one.*

$$\lim_{k\longrightarrow\infty} P\left(d_k(p_a, p_b) < \tau \mid p_a, p_b \in S_{\text{free}}\right) = 1. \tag{15}$$

Proof. The closed ball of radius $\tau > 0$ centred at $p_a \in T_a$ is defined as $O_a(p_a)$. Similarly, for $p_b \in T_b$ the closed ball is defined as $O_b(p_b)$.

$$O_a(p_a): O_{a,\tau} \coloneqq \left\{p \in S \mid \|p - p_a\| \le \tau\right\}, \tag{16}$$

$$O_b(p_b): O_{b,\tau} \coloneqq \left\{p \in S \mid \|p - p_b\| \le \tau\right\}. \tag{17}$$

Let $O'_a(p_a)$ denote the intersection of the obstacle-free space S_{free} and the closed ball $O_a(p_a)$. $O'_b(p_b)$ is denoted in the same way as $O'_a(p_a)$.

$$O'_a(p_a) = O_a(p_a) \cap S_{\text{free}}, \tag{18}$$

$$O'_b(p_b) = O_b(p_b) \cap S_{\text{free}}. \tag{19}$$

For $k = 0$, $d_0(p_a, p_b) = \|S_{\text{init}}, S_{\text{goal}}\|$, where $\|S_{\text{init}}, S_{\text{goal}}\|$ denotes the Euclidean distance from the initial to goal region. BPIB-RRT* guides two rapidly-exploring random trees towards each other by using the proposed *Bi-bias()* and *BPF()* heuristics. Therefore, the distance between T_a and T_b is decreasing.

$$P\{d_k(p_a, p_b) < \tau\} > 0. \tag{20}$$

For $k = n$, it is assumed that all vertices in T_a are outside the ball $O'_b(p_b)$.

$$T_a \cap O'_b(p_b) = \varnothing. \tag{21}$$

Similarly,

$$T_b \cap O'_a(p_a) = \varnothing. \tag{22}$$

For the next growth of the random tree T_a, according to the extension mode of BPIB-RRT*, if $q_{\text{op}} = p_b$, the mean of $D^a_{k+1}(p_b)$ is as shown below:

$$E\left[D^a_{k+1}(p_b) \mid q_{\text{op}} = p_b\right] < E\left[D^a_k(p_b)\right], \tag{23}$$

otherwise, the mean of $D^a_{k+1}(p_b)$ is given by following:

$$E\left[D^a_{k+1}(p_b) \mid q_{\text{op}} \ne p_b\right] \le E\left[D^a_k(p_b)\right]. \tag{24}$$

Thus,

$$E\left[D^a_{k+1}(p_b)\right] = E\Big[\alpha \bullet \left(D^a_{k+1}(p_b) \mid q_{op} = p_b\right) + (1-\alpha) \bullet \left(D^a_{k+1}(p_b) \mid q_{op} \ne p_b\right)\Big] < E\left[D^a_k(p_b)\right], \tag{25}$$

where $0 < \alpha < 1$.

Similarly, for the next growth of the random tree T_b, if $q_{\text{op}} = p_a$, then the mean of $D^b_{k+1}(p_a)$ is as given below:

$$E\left[D^b_{k+1}(p_a) \mid q_{\text{op}} = p_a\right] < E\left[D^b_k(p_a)\right]; \tag{26}$$

otherwise, the mean of $D^b_{k+1}(p_a)$ is expressed by the following form:

$$E\left[D^b_{k+1}(p_a) \mid q_{op} \ne p_a\right] \le E\left[D^b_k(p_a)\right]. \tag{27}$$

Therefore,

$$E\left[D^b_{k+1}(p_a)\right] = E\Big[\beta \bullet \left(D^b_{k+1}(p_a) \mid q_{op} = p_a\right) + (1-\beta) \bullet \left(D^b_{k+1}(p_a) \mid q_{op} \ne p_a\right)\Big] < E\left[D^b_k(p_a)\right], \tag{28}$$

where $0 < \beta < 1$.

Referring to (25) and (28), it is obvious that the distance between the vertex $p_a \in T_a$ and the vertex $p_b \in T_b$ is decreasing.

As the number of iterations k reaches infinity, the probability of finding a feasible path approaches one.

$$\lim_{k\longrightarrow\infty} P\left(d_k(p_a, p_b) < \tau \mid p_a, p_b \in S_{\text{free}}\right) = 1. \tag{29}$$

The BPIB-RRT* algorithm does not change the random sampling characteristic of the RRT* which ensures probabilistic completeness. □

5.2. Asymptotic Optimality. Given a cost function $J(\bullet)$, for an optimal path η^*, the optimal cost J^* can be confirmed by $J(\eta^*) = J^*$. Let $\eta^* \subset S_{\text{free}}$ denote the optimal path such that $\lim_{k\longrightarrow\infty}\eta_k = \eta^*$ and $\lim_{k\longrightarrow\infty} J(\eta_k) = J(\eta^*) = J^*$, where $\eta^* : [0,1] \longrightarrow \eta^*(0) = S_{\text{init}}, \eta^*(1) = S_{\text{goal}}$. Let $C_k^{\text{BPIB-RRT}^*}$ denote the random variable of the minimum path cost.

Theorem 2. *As the total number of iterations k reaches infinity, the cost of path generated by BPIB-RRT* converges to the optimal cost J^*.*

$$P\left(\left\{\lim_{k\longrightarrow\infty}\sup C_k^{\text{BPIB-RRT}^*} = J^*\right\}\right) = 1. \tag{30}$$

Sketch of Proof. In reference to Theorem 14 in [11], similar to RRT*, the BPIB-RRT* attempts to establish a connection between the roadmap vertexes within a ball of dynamic radius $r_k^{\text{BPIB-RRT}^*}$ that T_a and T_b can grow towards each other, where $r_k^{\text{BPIB-RRT}^*}$ is a function of k. The connection radius $r_k^{\text{BPIB-RRT}^*}$ is denoted as

$$r_k^{\text{BPIB-RRT}^*} = \gamma_{\text{BPIB-RRT}^*}\left(\frac{\log k}{k}\right)^{1/d}, \tag{31}$$

where

$\gamma_{\text{BPIB-RRT}^*} > \gamma^*_{\text{BPIB-RRT}^*} = 2(1 + (1/d))^{1/d}(\mu(S_{\text{free}})/\zeta_d)^{1/d}$.
$\mu(\bullet)$ denotes the Lebesgue measure, and ζ_d denotes the volume of the unit ball in the d-dimensional Euclidean space. As the same procedures happened in BPIB-RRT*, according to Lemma 56, 71, and 72 in [12], it can be proved that BPIB-RRT* is asymptotically optimal.

5.3. Computational Complexity. Let $M_k^{\text{BPIB-RRT}^*}$ denote the mean of the total computation performed by BPIB-RRT* with k iterations. Theorem 3 proposes that the computational complexity of BPIB-RRT* is a constant multiple higher than IB-RRT*.

Theorem 3. *There exists a constant $\phi \in R_+$ such that it satisfies the following formula:*

$$\lim_{k\longrightarrow\infty}\sup E\left[\frac{M_k^{\text{BPIB-RRT}^*}}{M_k^{\text{IB-RRT}^*}}\right] \le \phi. \tag{32}$$

Sketch of Proof. Compared to IB-RRT*, *Bi-bias()* and *BPF()* heuristics are introduced into BPIB-RRT*. Notice that *Bi-bias()* and *BPF()* heuristics are both performed in a constant number of iterations and independent of the number of vertices in the tree. *BPF()* has to calculate all obstacles for the bias point $q_{\text{bia}} \in S_{\text{free}}$ which requires at least $\Omega(\log_{10} n)$ time. Furthermore, BPIB-RRT* calls *NearestVertex*(), *NeighbouringVertices(), Steer()*, and *ObstacleFree*() procedures for both trees T_a and T_b just like IB-RRT* which adds up a computation constant. Hence, as seen in Theorem 3, BPIB-RRT* only vary from IB-RRT* by ϕ in terms of computational complexity ratio.

6. Simulation Results

In order to analyse the effectiveness of BPIB-RRT*, the 3D simulation examples will be made for three different 3D configuration spaces in this section.

Three different 3D configuration spaces are shown in Figure 3. The initial and goal locations are fixed at $(0,0,0)$ and $(100,100,100)$, respectively in Map1 and Map2. The initial and goal locations are fixed at $(0,0,5)$ and $(100,100,5)$ in Map3. Spherical obstacles are generated in the configuration spaces to examine the efficiency of our proposed BPIB-RRT* algorithm.

The maximum number of algorithm iterations is set as 10^6. If the total number of iterations exceeds the maximum, the simulation returns a feasible path or a failed one. BPIB-RRT*, PIB-RRT*, tGSRT, and RRT* are examined in three aforementioned different 3D environments, and the simulation results are shown in Figures 4–9. From Figures 4–9, it can be obtained that PIB-RRT*, tGSRT, and BPIB-RRT* are effective for UAV path planning in 3D environments, whereas the traditional RRT* algorithm may be failed. Besides, all the paths generated by our proposed BPIB-RRT* algorithm can avoid obstacles, which have been obviously shown in Figures 7–9 with different visual angles. Table 1 shows the parameters obtained while running 50 times of the four different algorithms with maximum, minimum, and average iterations and path cost (in terms of Euclidean metric). The conclusions of the examination and the simulations are discussed in detail below.

Figure 4 shows the simulation results of RRT* in three different 3D maps. As seen from Table 1, RRT* performs pure exploration that too many invalid samples lower efficiency of the algorithm. It may be hard or need a lot of iterations for RRT* to find an optimal path. Therefore, a longer planning time is necessary for RRT* to converge to an optimal solution. Meanwhile, RRT* is failed to converge to the optimal solution 76%, 70%, and 62% of the times in three different maps, respectively.

The path planning results in three different environments for PIB-RRT* are illustrated in Figure 5. Compared with the RRT* algorithm, the PIB-RRT* algorithm converges to the optimal solution faster. Random trees are potentially guided by bidirectional potential gradient heuristics towards each other to improve the sampling efficiency and reduces the planning time of the PIB-RRT* in three different maps (Time = 25.627, 23.633, 25.739). However, the fixed attractive pole inside the *BPG()* heuristic still produced invalid sampling points. Also, the smoothness of the path is not considered in PIB-RRT*.

It can be seen from Figure 6 that the tGSRT algorithm can search the path effectively in three different maps. However, the path generated by tGSRT algorithm with jags which cannot meet the constraint conditions of smooth and the jags may cause unstability for the flight of UAV. Furthermore, compared with the other algorithms, the paths generated by the tGSRT algorithm in three different maps are not the optimal solution (Path Cost = 204.861, 205.797, 195.709). Meanwhile, the planning time is increased (Time = 54.982, 60.663, 45.422) due to the evolutionary

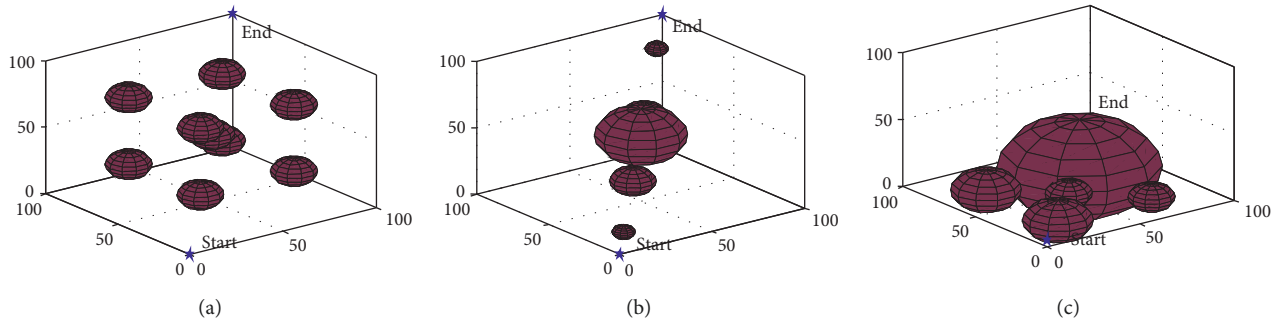

Figure 3: 3D configuration spaces. (a) Map1, (b) Map2, and (c) Map3.

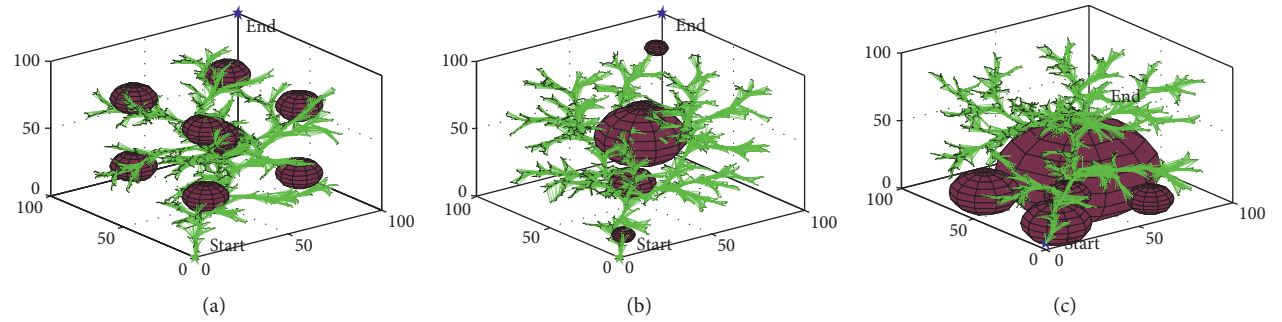

Figure 4: Simulation results of RRT* in three different 3D environments. (a) Map1, (b) Map2, and (c) Map3.

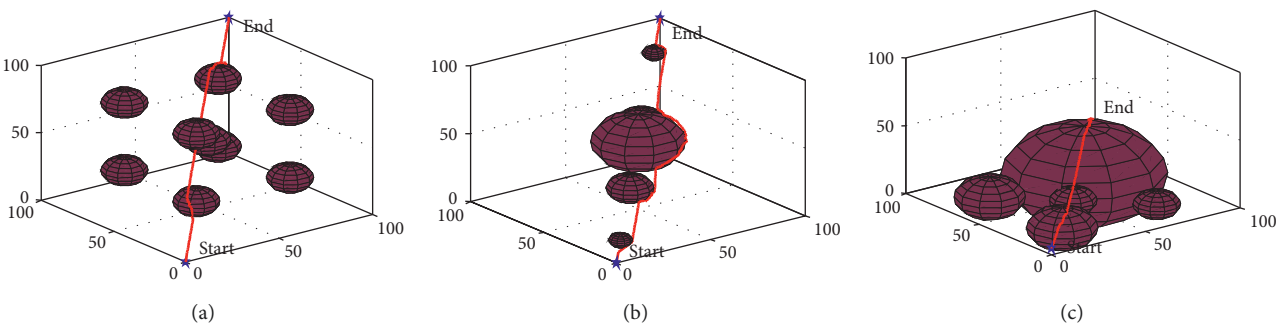

Figure 5: Simulation results of PIB-RRT* in three different 3D environments. (a) Map1, (b) Map2, and (c) Map3.

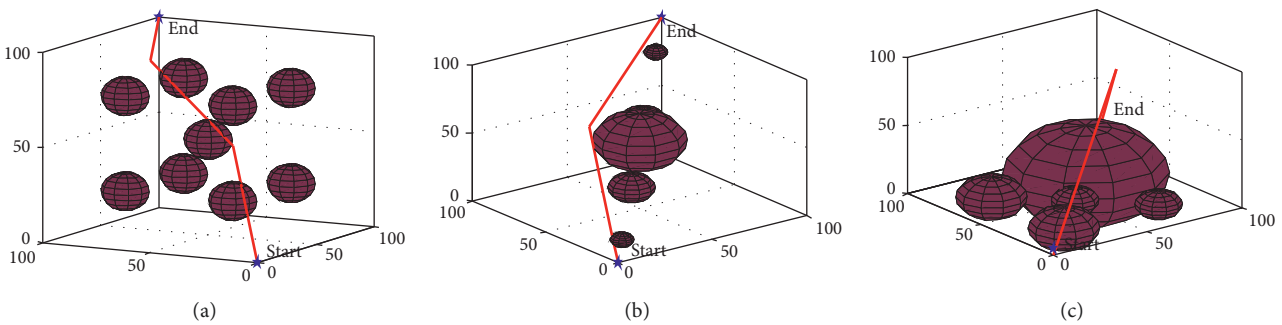

Figure 6: Simulation results of tGSRT in three different 3D environments. (a) Map1, (b) Map2, and (c) Map3.

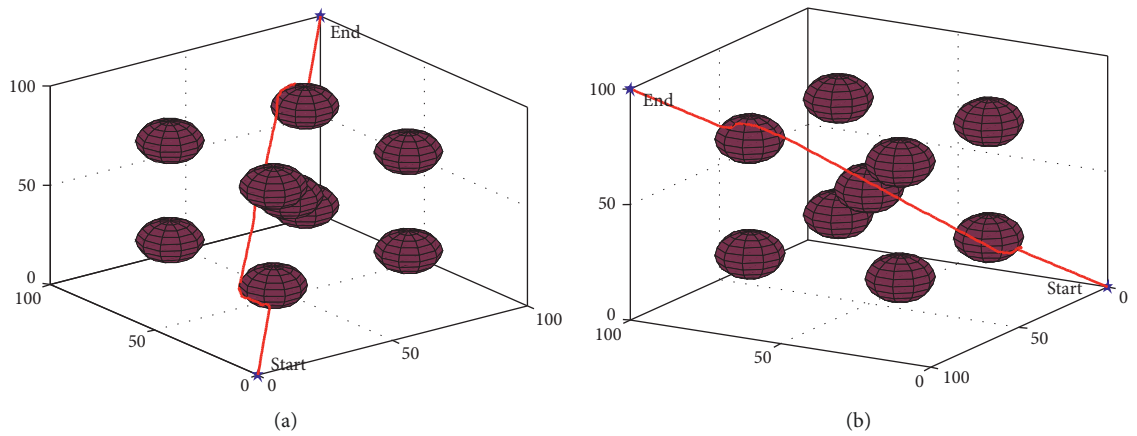

Figure 7: Simulation results of BPIB-RRT* in Map1. (a) AZ = −37.5, EL = 30, and (b) AZ = −150, EL = 12.

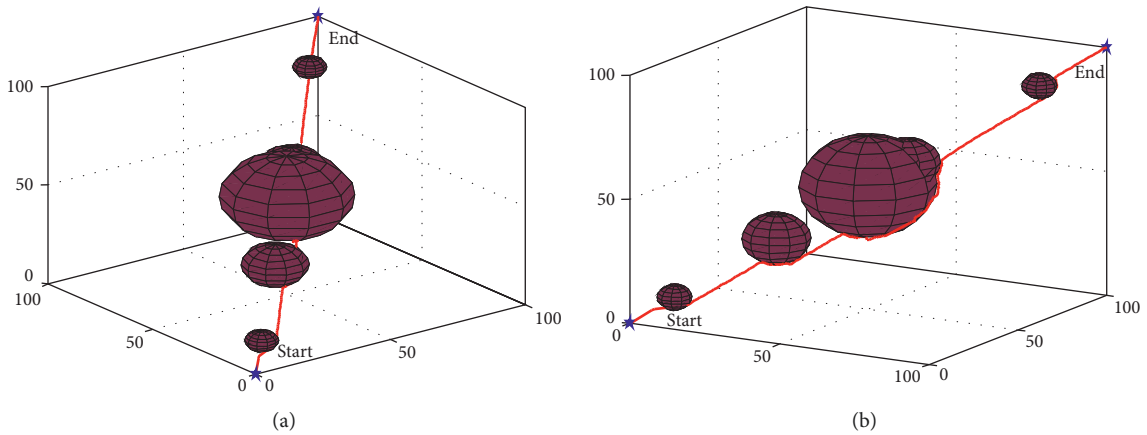

Figure 8: Simulation results of BPIB-RRT* in Map2. (a) AZ = −37.5, EL = 30, and (b) AZ = 31, EL = 18.

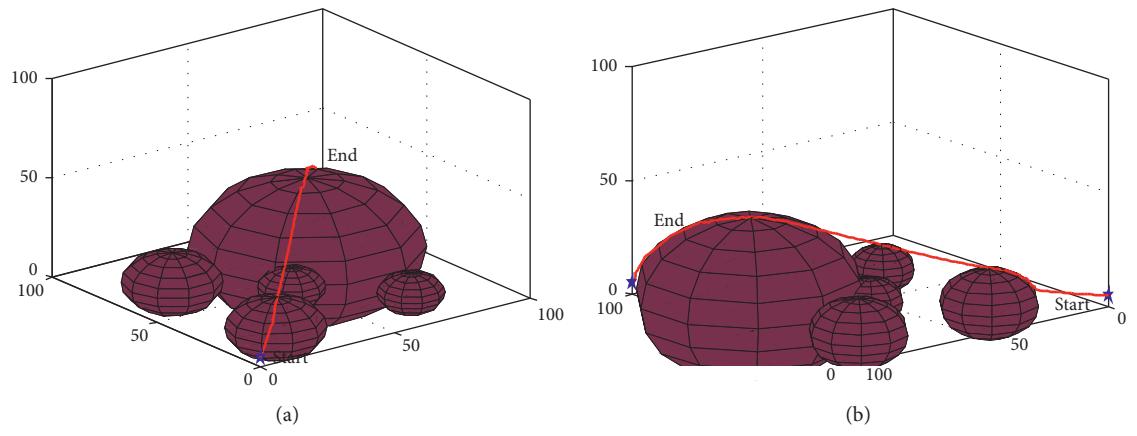

Figure 9: Simulation results of BPIB-RRT* in Map3. (a) AZ = −37.5, EL = 30, and (b) AZ = −130, EL = 22.

process (selection, crossover and mutation) of the genetic algorithm.

The simulation results in three maps of BPIB-RRT* are shown in Figures 7–9. Also, different visual angle of results are given to present obviously. As seen from Table 1, based on the *Bi-bias()* heuristic and *BPF()* heuristic, the optimal solution is found in the Map1, most quickly with more efficient sampling strategy by BPIB-RRT*(Time = 8.532 s). Comparing with PIB-RRT*(Average Iterations = 205.473), the average number of iterations of BPIB-RRT* is reduced (Average Iterations = 104.368). To improve the flight ability of the BPIB-RRT*-generated path, the path vertices

Table 1: Simulation result data.

Environment	Algorithm	Path cost (km)	Max iteration	Min iteration	Average iteration	Time (s)	Failed rate (%)
Map1	RRT*	203	15607	6870	13920.768	680.453	76
	B-RRT*	203	780	389	509.447	30.129	
	PIB-RRT*	203	231	192	205.473	25.627	—
	tGSRT	204.861	316	121	198.65	54.982	—
	BPIB-RRT*	203	129	98	104.368	8.532	—
Map2	RRT*	202	15893	7378	12685.151	684.434	70
	B-RRT*	202	732	387	420.196	33.647	
	PIB-RRT*	202	221	179	194.473	23.633	—
	tGSRT	205.797	433	192	278.35	60.663	—
	BPIB-RRT*	202	199	168	179.947	19.537	—
Map3	RRT*	168.656	17428	12453	15527.745	862.011	62
	B-RRT*	168.656	621	389	453.169	34.551	
	PIB-RRT*	168.656	251	156	204.483	25.739	—
	tGSRT	195.709	754	636	705.974	45.422	—
	BPIB-RRT*	168.656	178	164	169.427	17.018	—

sequence is fed into B-spline to robustly produce a smooth optimal path. Figure 8 is another 3D environment with obstacles gradually growing larger, and BPIB-RRT* is the first to find the optimal path(Time = 19.537 s), whereas PIB-RRT* and B-RRT* are the second (Time = 23.633 s) and the third (Time = 33.647 s) converge to the optimal path. Figure 9 is the results of Map3, and UAV can smoothly pass through the obstacles. As seen from Table 1, BPIB-RRT* is the fastest to find the optimal path (Time = 17.018), comparing with PIB-RRT* (Time = 25.739) and RRT* (Time = 862.011).

Based on aforementioned analyses, it can be known that the proposed BPIB-RRT* algorithm can solve the path planning problem for UAV in complex 3D environment and avoid obstacle with excellent performance. Moreover, comparing with other variants of the RRT* algorithm, the average number of iterations is minimal and the convergence rate is the fastest by using the BPIB-RRT* algorithm.

7. Conclusions

In this paper, a new sampling-based approach defined as the BPIB-RRT* algorithm is proposed, which can obtain the optimal path in the complex 3D configuration space. Higher deviation can be provided by the *Bi-bias()* heuristic which can adjust the space of the sample and improve the accuracy of it. The *BPF()* heuristic is introduced into the BPIB-RRT* algorithm to guide the path node by the resultant force based on the goal node and the obstacle and reduce the blindness of the algorithm which expedites the convergence rate. Also, the iteration number of the BPIB-RRT* algorithm can be reduced benefited by the aforementioned heuristics. Finally, the curvature of path turns into continuous by B-spline which guarantees the stability of UAV. Meanwhile, it is strictly proven that as the number of iteration reaches infinity, the probability of finding a feasible solution is limited to 1. Computational complexity has been testified to be both theoretically and experimentally equal to the IB-RRT* algorithm, and the presented algorithm inherits the asymptotic optimality of RRT*. Comparing with RRT* and other existing algorithms, the superiority of the proposed BPIB-RRT* algorithm has been verified in the comparison experiments.

Data Availability

The data used to support the findings of this study are available from the corresponding author upon request.

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this paper.

Acknowledgments

This paper was partially supported by the National Defense Basic Scientific Research Project JC*** and Higher School Science and Technology Research Project of Hebei Province (BJ2017041).

References

[1] A. A. Doshi, A. J. Postula, A. Fletcher, and S. P. N. Singh, "Development of micro-UAV with integrated motion planning for open-cut mining surveillance," *Microprocessors and Microsystems*, vol. 39, no. 8, pp. 829–835, 2015.

[2] M. Półka, P. Szymon, and K. Łukasz, "The use of UAV's for search and rescue operations," *Procedia Engineering*, vol. 192, pp. 748–752, 2017.

[3] G. C. Lin, Y. C. Tang, X. J. Zou et al., "Color-, depth-, and shape-based 3D fruit detection," *Precision Agriculture*, 2019.

[4] K. Ivushkin, H. Bartholomeus, A. K. Bregt et al., "UAV based soil salinity assessment of cropland," *Geoderma*, vol. 338, pp. 502–512, 2019.

[5] X. Cao, X. Zou, C. Jia, M. Chen, and Z. Zeng, "RRT-based path planning for an intelligent litchi-picking manipulator," *Computers and Electronics in Agriculture*, vol. 156, pp. 105–118, 2019.

[6] N. Chao, Y.-K. Liu, H. Xia, A. Ayodeji, and L. Bai, "Grid-based RRT* for minimum dose walking path-planning in complex radioactive environments," *Annals of Nuclear Energy*, vol. 115, pp. 73–82, 2018.

[7] H. Wu, H. Li, R. Xiao, and J. Liu, "Modeling and simulation of dynamic ant colony's labor division for task allocation of UAV swarm," *Physica A: Statistical Mechanics and Its Applications*, vol. 491, pp. 127–141, 2018.

[8] X. Wu, W. Bai, Y. Xie, X. Sun, C. Deng, and H. Cui, "A hybrid algorithm of particle swarm optimization, metropolis criterion and RTS smoother for path planning of UAVs," *Applied Soft Computing*, vol. 73, pp. 735–747, 2018.

[9] L. P. Behnck, D. Doering, C. E. Pereira, and A. Rettberg, "A modified simulated annealing algorithm for SUAVs path planning," *IFAC-PapersOnLine*, vol. 48, no. 10, pp. 63–68, 2015.

[10] A. Abbaspour, K. K. Yen, S. Noei, and A. Sargolzaei, "Detection of fault data injection attack on UAV using adaptive neural network," *Procedia Computer Science*, vol. 95, pp. 193–200, 2016.

[11] S. Karaman and E. Frazzoli, "Incremental sampling-based algorithms for optimal motion planning," 2010, https://arxiv.org/abs/1005.0416.

[12] S. Karaman and E. Frazzoli, "Sampling-based algorithms for optimal motion planning," *The International Journal of Robotics Research*, vol. 30, no. 7, pp. 846–894, 2011.

[13] N. Chao, Y. K. Liu, H. Xia, M. J. Peng, and A. Ayodeji, "DL-RRT* algorithm for least dose path re-planning in dynamic radioactive environments," *Nuclear Engineering and Technology*, vol. 51, no. 3, pp. 825–836, 2019.

[14] J. J. Kuffner and S. M. LaValle, "RRT-connect: an efficient approach to single-query path planning," in *Proceedings of the 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation: Symposia Proceedings (Cat. No. 00CH37065)*, Piscataway, NJ, USA, 2000.

[15] A. Hidalgo-Paniagua, J. P. Bandera, M. Ruiz-De-Quintanilla, and A. Bandera, "Quad-RRT: a real-time GPU-based global path planner in large-scale real environments," *Expert Systems with Applications*, vol. 99, no. 1, pp. 141–154, 2018.

[16] M. Jordan and A. Perez, "Optimal bidirectional rapidly-exploring random trees," Tech. Rep. MIT-CSAIL-TR-2013-021, Computer Science and Artificial Intelligence Laboratory, Massachusetts Institute of Technology, Cambridge, MA, USA, 2013.

[17] D. Lee and D. H. Shim, "Path planner based on bidirectional spline-RRT* for fixed-wing UAVs," in *Proceedings of the 2016 International Conference on Unmanned Aircraft Systems (ICUAS)*, June 2016.

[18] A. H. Qureshi and Y. Ayaz, "Intelligent bidirectional rapidly-exploring random trees for optimal motion planning in complex cluttered environments," *Robotics and Autonomous Systems*, vol. 68, pp. 1–11, 2015.

[19] A. H. Qureshi and Y. Ayaz, "Potential functions based sampling heuristic for optimal path planning," *Autonomous Robots*, vol. 40, no. 6, pp. 1079–1093, 2016.

[20] Z. Tahir, A. H. Qureshi, Y. Ayaz, and R. Nawaz, "Potentially guided bidirectionalized RRT* for fast optimal path planning in cluttered environments," *Robotics and Autonomous Systems*, vol. 108, pp. 13–27, 2018.

[21] J. Sun, J. Tang, and S. Lao, "Collision avoidance for cooperative UAVs with optimized artificial potential field algorithm," *IEEE Access*, vol. 5, pp. 18382–18390, 2017.

[22] Z. Zhou, J. Wang, Z. Zhu, D. Yang, and J. Wu, "Tangent navigated robot path planning strategy using particle swarm optimized artificial potential field," *Optik—International Journal for Light and Electron Optics*, vol. 158, pp. 639–651, 2018.

[23] W. G. Li, S. Y. Sun, J. Z. Li et al., "UAV dynamic path planning algorithm based on segmentated optimization RRT," *Systems Engineering and Electronics*, vol. 40, no. 8, pp. 123–130, 2018.