Research Article

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

Xiaojing Wu1,2, Lei Xu1, Ran Zhen1,2, and Xueli Wu1,2

1School of Electrical Engineering, Hebei University of Science and Technology, Shijiazhuang, Hebei 050018, China
2Hebei 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

Copyright © 2019 Xiaojing Wu et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

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 Bidirectional-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 Bidirectional 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 bidirectional 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 $S \subset \mathbb{R}^3$. Furthermore, the given configuration space is divided into obstacle regions and obstacle-free regions which are denoted by $S_{obs} \subset S$ and $S_{free} = S \setminus S_{obs}$, respectively. Let $T_a = (V_a, E_a) \subset S_{free}$ and $T_b = (V_b, E_b) \subset S_{free}$ represent two random trees which are initialized by $S_{init} \Subset S_{free}$ and $S_{goal} \Subset S_{free}$ 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 $\{\eta_n\} \Subset S_{free}$ 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 $\eta[0,1] \rightarrow \{\eta(0) = S_{init}, \eta(1) = S_{goal}\} \subset S_{free}$.

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 $n \rightarrow N$. The cost function $J$ can be defined as...
\[ J = \sum_{n=1}^{N} L(n), \]  

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

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 \( s_{\text{init}} \) to start the iterative process (Line 1). The random sample \( q_{\text{rand}} \in S \) is given by \textit{Sample} () (Line 3). Then, the near vertices or the nearest vertex of \( q_{\text{rand}} \) is obtained by \textit{NearVertices}() or \textit{NearestVertex}() (Line 4–7). The populated set \( Q_{\text{near}} \) is then sorted by the \textit{GetSortedList()} procedure (Line 8). \textit{Choose-BestParent()} returns \( q_{\text{min}} \in Q_{\text{near}} \) with minimum cost from the random tree \( T \) to \( q_{\text{rand}} \) in obstacle-free space (Line 9). Once the best parent vertex is located, \( d_{\text{min}} \) is inserted into the random tree \( T \) by making \( q_{\text{rand}} \) its child and the rewiring procedure is executed (Line 10–13). Algorithm iteration continues till \( k \rightarrow K \) 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 \( S_{\text{near}} \) and the same process is acted on this random node \( q_{\text{rand}} \) (Line 4–9). Then, \( q_{\text{rand}} \) is inserted into the random tree \( T_a \) by \textit{InsertVertex}() (Line 10–13), and the nearest node \( q_{\text{conn}} \) from tree \( T_b \) is found (Line 14). If \textit{Connect()} heuristic successfully connects random trees \( T_a \) and \( T_b \), a feasible path \( d_{\text{new}} \) from \( s_{\text{init}} \) to \( s_{\text{goal}} \) is generated (Line 15). The cost of \( d_{\text{new}} \) is compared with the previously computed path cost \( d_{\text{act}} \) 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 \textit{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 \( q_{\text{rand}} \) is produced by \textit{Sample}(). Then, the near neighbour vertices \( Q_{\text{near}}^a \) and \( Q_{\text{near}}^b \) are generated by \textit{NeighboringVertices}() from \( T_a \) and \( T_b \), respectively. Both neighbouring vertex sets \( Q_{\text{near}}^a \) and \( Q_{\text{near}}^b \) are sorted by the \textit{GetSortedList()} procedure. The \textit{GetBestTreeParent()} heuristic returns the best parent \( P_{\text{min}} \) from \( T_a \) or \( T_b \). Then, the \( q_{\text{rand}} \) 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 \textit{BPG()} heuristic. The \textit{BPG()} heuristic returns the potentially guided bidirectional randomly sampled state \( q_{\text{pb}} \). The \textit{BPG()} heuristic is shown in Algorithm 3 and explained below.

\textit{BPG()} if the iteration number \( k \) is even, the potentially guided bidirectional randomly sampled state \( q_{\text{pb}} \) is fed into the \textit{BPGgoal()} heuristic, which is used to calculate the attractive force \( F_{\text{att}} \) working on \( q_{\text{pb}} \). Comparing \( d'_{\text{near}} \) with a constant \( d'_{\text{bias}} \), \( q_{\text{pb}} \) will be guided downhill in the direction of decreasing potential towards the goal region. Similarly, for odd iteration \( k \), \( q_{\text{pb}} \) is passed to \textit{BPGinit()} heuristic and attractive force \( F_{\text{att}} \) is calculated. Then, \( q_{\text{pb}} \) is pulled towards \( S_{\text{init}} \) by the \textit{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\(^*\) (PIB-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 PIB-RRT\(^*\) algorithm can be presented in Figure 1. Extra procedures employed by PIB-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\(^*\) \( q_{\text{rand}} \) is generated by \textit{Sample}() firstly and then passed to \textit{Bi-bias}(). If the iteration count \( k \) is even, then the randomly sampled state \( q_{\text{rand}} \) is pulled toward the random tree \( T_b \). The bias direction can be obtained by analysing the relative position between \( q_{\text{prev1}} \) and \( q_{\text{rand}} \), where \( q_{\text{prev1}} \) 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 \( q_{\text{bias}} \). If the iteration count \( k \) is odd, the random tree \( T_a \) attracts the randomly sampled state \( q_{\text{rand}} \). Similarly, the relative position between \( q_{\text{prev2}} \) and \( q_{\text{rand}} \) is calculated to provide a higher deviation for \( q_{\text{rand}} \), where \( q_{\text{prev2}} \) is the new vertex of the target tree \( T_a \).

The bias formulas are shown below:
4 Mathematical Problems in Engineering

Algorithm 1: RRT⁺ (S₀goal).

Algorithm 2: B-RRT⁺ (S₀goal).

Algorithm 3: BPG(qrand, k).

The mathematical problems in this document are related to optimization and control systems, specifically focusing on path planning and robotic navigation. The text provides algorithms for RRT⁺ and B-RRT⁺, which are probabilistic roadmaps used for motion planning in robotics. The BPG algorithm is also discussed, which is a biased potential field approach for improving sampling efficiency.

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 toward it, and the obstacles repel the robot away from the obstacle configuration space Sobs. 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 tendency and the velocity are always continuous [22] and improves UAV stability.

The equation of attraction field is discussed below:

\[ U_A (q) = \frac{1}{2} k_A \rho^2 (q, S_{goal}) \]  

where \( U_A (q) \) is the attraction field, \( k_A \) is the attraction field coefficient, \( q \) is the current position of the UAV, \( S_{goal} \) is the goal location, and \( \rho(q, S_{goal}) \) is the distance between the current point and the goal location. The attraction field force \( F_A(q) \) is generated by taking a negative gradient.
\[
\begin{align*}
\mathcal{F}_{\text{Att}}(q) &= -\nabla U_{\text{Att}}(q) = k_{\text{Att}}B(q, S_{\text{goal}}) . \quad (7)
\end{align*}
\]

The repulsive field \( U_{\text{Rep}}(q) \) is generated by all obstacles
\( S_{\text{obs}} \in S \).
\[
U_{\text{Rep}}(q) = \sum_{i=1}^{I} U_{\text{Rep}}^i(q),
\]
\[
U_{\text{Rep}}^i(q) = \begin{cases} 
\frac{1}{2} \left( \frac{1}{\rho(q, q_{i_{\text{obs}}})} - \frac{1}{\rho_{0}^i} \right)^2, & \rho(q, q_{i_{\text{obs}}}) < \rho_{0}^i, \\
0, & \rho(q, q_{i_{\text{obs}}}) \geq \rho_{0}^i,
\end{cases}
\]
where \( U_{\text{Rep}}^i(q) \) is the repulsion field generated by the \( i \)-th obstacle at \( q \).

\[
\rho_{0}^i = \frac{2}{\lambda} \left( \frac{1}{\rho(q, q_{i_{\text{obs}}})} - \frac{1}{\rho_{0}^i} \right)^2, \quad \rho(q, q_{i_{\text{obs}}}) < \rho_{0}^i,
\]

where \( \lambda \) is the repulsive field coefficient, \( \rho_{0}^i \) is the effective
radius of the \( i \)-th repulsive field, and \( \rho(q, q_{i_{\text{obs}}}) \) is the distance
between the \( i \)-th obstacle and the current position \( q \). Correspondingly, let \( F_{\text{Rep}}^i(q) \) sign the repulsion force,
\[
F_{\text{Rep}}^i(q) = \begin{cases} 
F_{\text{Rep1}}(q), & \rho(q, q_{i_{\text{obs}}}) < \rho_{0}^i, \\
0, & \rho(q, q_{i_{\text{obs}}}) \geq \rho_{0}^i,
\end{cases}
\]
where \( F_{\text{Rep1}}(q) = \lambda ((1/\rho(q, q_{i_{\text{obs}}})) - (1/\rho_{0}^i)) (1/\rho^2(q, q_{i_{\text{obs}}})) \nabla \rho(q, q_{i_{\text{obs}}}).\)

The resultant force on the UAV at point \( q \) is \( F(q) \).
\[
F(q) = \mathcal{F}_{\text{Att}}(q) + \sum_{i=1}^{I} F_{\text{Rep}}^i(q) .
\]

Incorporating APF into IB-RRT* by using the Bi-directional Potential Field heuristic \( (BPF()) \), pseudocode of the \( BPF() \) heuristic is presented in Algorithm 6. The current vertices of two random trees \( q_{\text{prev1}} \) and \( q_{\text{prev2}} \) are first fed into the \( BPF() \) algorithm. When the number of iterations \( k \) is even, the new vertex \( q_{\text{prev2}} \) of the \( T_b \) is treated as the target point of the bias point \( q_{\text{bias}} \), and the attraction force based on
the $q_{prev2}$ is calculated. $d_{nearest}$ is the distance between the bias point $q_{bias}$ and the $i$th obstacle. $r_0^*$ denotes the radius of repulsion field. If $d_{nearest}$ is less than $r_0^*$, $q_{bias}$ 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 $F_{attr}(q)$ works on $q_{bias}$. Then, the optimal random node $q_{opt}$ 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 $q_{bias}$. Similarly, $q_{bias}$ is directed downhill in direction of the decreasing potential by computing the resultant force, and the optimal random node $q_{opt}$ 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:

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

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

$$F_{0,3}(t) = \frac{1}{6} (1 - t)^3,$$
$$F_{1,3}(t) = \frac{1}{6} (3t^3 - 6t^2 + 4),$$
$$F_{2,3}(t) = \frac{1}{6} (-3t^3 + 3t^2 + 3t + 1),$$
$$F_{3,3}(t) = \frac{1}{6} t^3,$$

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

5. Analysis

5.1. Probabilistic Completeness. Let $D_{p}(x)$ denote the random variable of the distance with the minimum cost from node $p \in S_{tree}$ to the first random tree $T_a$, and $d_k^e$ is the value of the random variable $D_{p}(x)$, where $k$ is the number of iterations for BPIB-RRT*. Similarly, let $D_k(p_a, p_b)$ denote the random variable of the minimum distance cost between nodes $p \in S_{tree}$ and $T_b$, and $d_k^e$ 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. $d_k^e$ 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
Algorithm 6: BPF ($q_{bias}$, $k$, $q_{prev1}$, $q_{prev2}$).

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

$$\lim_{k \to \infty} P(d_k(p_a, p_b) < \tau \mid p_a, p_b \in S_{free}) = 1.$$ (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.

$$\lim_{k \to \infty} P(d_k(p_a, p_b) < \tau \mid p_a, p_b \in S_{free}) = 1.$$ (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 $O_b(p_b)$.

$$O_a(p_a) : O_{a, r} = \{ p \in S \mid \| p - p_a \| \leq \tau \},$$ (16)

$$O_b(p_b) : O_{b, r} = \{ p \in S \mid \| p - p_b \| \leq \tau \}. $$ (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 $O'_a(p_a)$.

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

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

For $k = 0$, $d_k(p_a, p_b) = \| S_{init} - S_{goal} \|$, where $\| S_{init} - S_{goal} \|$ 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.

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

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

$$T_a \cap O'_b(p_b) = \emptyset.$$ (21)

Similarly,

$$T_b \cap O'_a(p_a) = \emptyset.$$ (22)

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

$$E[D^a_{k+1}(p_b) \mid q_{op} = p_b] < E[D^a_k(p_b)].$$ (23)

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

$$E[D^a_{k+1}(p_b) \mid q_{op} \neq p_b] < E[D^a_k(p_b)].$$ (24)

Thus,

$$E[D^a_{k+1}(p_b)] = E[a \cdot (D^a_{k+1}(p_b) \mid q_{op} = p_b)] + (1 - a) \cdot (D^a_{k+1}(p_b) \mid q_{op} \neq p_b) < E[D^a_k(p_b)],$$ (25)

where $0 < a < 1$.

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

$$E[D^b_{k+1}(p_a) \mid q_{op} = p_a] < E[D^b_k(p_a)].$$ (26)

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

$$E[D^b_{k+1}(p_a) \mid q_{op} \neq p_a] < E[D^b_k(p_a)].$$ (27)

Therefore,

$$E[D^b_{k+1}(p_a)] = E[\beta \cdot (D^b_{k+1}(p_a) \mid q_{op} = p_a)] + (1 - \beta) \cdot (D^b_{k+1}(p_a) \mid q_{op} \neq p_a) < E[D^b_k(p_a)],$$ (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.

$$\lim_{k \to \infty} P(d_k(p_a, p_b) < \tau \mid p_a, p_b \in S_{free}) = 1.$$ (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 \( \eta^* \), the optimal cost \( J^* \) can be confirmed by \( J(\eta^*) = J^* \). Let \( \eta^* \subset S_{\text{free}} \) denote the optimal path such that \( \lim_{k \to +\infty} \eta_k = \eta^* \) and \( \lim_{k \to +\infty} J(\eta_k) = J(\eta^*) = J^* \), where \( \eta^* : [0, 1] \to \eta^*(0) = S_{\text{init}}, \eta^*(1) = S_{\text{goal}} \). Let \( C_{BPIB-RRT^*} \) 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 \( J^* \).

\[ P\left\{ \lim_{k \to +\infty} \sup_{C_{BPIB-RRT^*}} = J^* \right\} = 1. \] (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 \( r_{BPIB-RRT^*} \) that \( T_a \) and \( T_b \) can grow towards each other, where \( r_{BPIB-RRT^*} \) is a function of \( k \). The connection radius \( r_{BPIB-RRT^*} \) is denoted as

\[ r_{BPIB-RRT^*} = y_{BPIB-RRT^*} \left( \log k \right)^{1/d}, \] (31)

where \( y_{BPIB-RRT^*} > y_{BPIB-RRT^*} = 2 \left( 1 + (1/d) \right) \mu(S_{\text{free}}))^{1/d}. \mu(\bullet) \) 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 \( M_{BPIB-RRT^*} \) 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:

\[ \lim_{k \to +\infty} \sup \left\{ \frac{M_{BPIB-RRT^*}}{M_{IB-RRT^*}} \right\} \leq \phi. \] (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 vertexes in the tree. BPF() has to calculate all obstacles for the bias point \( q_{\text{bias}} \in S_{\text{free}} \) which requires at least \( \Omega \left( \log_{10} n \right) \) time. Furthermore, BPIB-RRT* calls NearestVertex(), NeighbouringVertex(), 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 \( \phi \) 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.
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 angles 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 \( = 8.532 \text{ s} \)). 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

\[ \text{Figure 7: Simulation results of BPIB-RRT* in Map1. (a) AZ = -37.5, EL = 30, and (b) AZ = -150, EL = 12.} \]

\[ \text{Figure 8: Simulation results of BPIB-RRT* in Map2. (a) AZ = -37.5, EL = 30, and (b) AZ = 31, EL = 18.} \]

\[ \text{Figure 9: Simulation results of BPIB-RRT* in Map3. (a) AZ = -37.5, EL = 30, and (b) AZ = -130, EL = 22.} \]
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 = 25.739 s), comparing with PIB-RRT* (Time = 25.739) and RRT* (Time = 862.011 s).

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 obstacles 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 (JCKY201701030) and Higher School Science and Technology Research Project of Hebei Province (B201701041).

References


