Path planning for UAV based on improved dynamic step RRT algorithm

Jing Xiong1 and Xiaokun Duan
Changzhou College of Information Technology, Changzhou, China

1 E-mail: xiongjwh@163.com

Abstract. Path planning is an important problem in the field of unmanned aerial vehicles (UAVs). However, it is inefficient for many rapidly-exploring random tree (RRT) based methods to rapidly find a feasible solution in a complex environment. To solve the path planning problem for the UAV in a complex environment, an improved dynamic step size RRT algorithm combined with a new path length control strategy is proposed. Firstly, the algorithm adopts a biased-goal sampling strategy to guide the growth of the tree, and an expansion direction constraint strategy is adopted to limit the expansion direction of the tree. Secondly, an improved dynamic step size strategy is proposed to speed up the path searching process. Thirdly, a path length constraint strategy is designed, which is used to constrain the path length during the searching process for a solution. Finally, simulation results demonstrate that the proposed method achieves improvement in both computational time and the path quality.

1. Introduction
Path planning is one of the most fundamental research topics in the field of unmanned aerial vehicles (UAVs). The objective of UAV path planning is to compute a valid and feasible solution that connects a start point and a goal point to achieve a given task in a certain environment. Meanwhile, UAV path planning with different requirements has been widely studied.

Due to the influence of obstacles, especially in a complex environment, UAVs must have the ability to rapidly plan paths to fly safely. In study [1-3], methods based on geometric approaches such as A* algorithm have been proposed. A* algorithm searches for optimal solutions since it formulates the planning task as a path cost minimization problem. Moreover, these kind methods are strongly based on the cost map, and it is a very time-consuming task to produce and store the cost map, particularly in 3D implementation [4]. Computational intelligence (CI) is a kind of nature-inspired computational approach to solve complex optimization problems. Popular and efficient CI approaches include such as genetic algorithm (GA), particle swarm optimization (PSO), ant colony optimization (ACO), and so on [5-9]. Hot issues in CI approach research are to solve problems of slow convergence speed, low searching efficiency, and falling into the local minima [10-12].

Rapidly-exploring random tree (RRT) is a sampling-based algorithm, and it is firstly introduced in Literature [13]. RRT is capable of most motion planning problems, and it effectively handles problems with different constraints [14]. RRT method has the advantage that it can generate a feasible path relatively quickly and avoid the difficulty of modeling [15]. Therefore, RRT and its variants have a wide research and application in UAV, robot, and other path planning fields. However, basic RRT searches completely random. As a result, it consumes lots of time due to many iterations utilized in exploring areas which are unnecessary to explore. For UAV path planning, especially in real-time
applications, the rapidity of planning is almost the most important requirement since the UAV must respond rapidly to avoid crash. To accelerate the planning process, many variants have been proposed, such as efficient bias-goal factor RRT (EBG-RRT) [16], Multiple RRT [17-19], dynamic step RRT [20-22], and so on. Literature [20] adapts the step size of RRT based on distance from the root node, and as the RRT tree grows, the step size will get larger. However, setting a large step size is not a good choice for solving path planning problems in a complex environment, because it will cause collisions, failure of the tree to grow, and increase in solution time. Literature [21] proposes to adjust the step size of the expanding tree according to the distance from a randomly selected node, and after an initial path is generated, then it tries to merge the nodes in the path to optimize the path length. This method performance well in obtaining an initial path quickly. However, these strategies have failed to consider the growth trend of the tree, this may cause the tree growing away from the target area with a large step size and slowing down the convergence speed.

This work aims to research the UAV path planning problem in a complex environment. In this work, an improved RRT algorithm is proposed. It introduces a new dynamic step size strategy which provides different step sizes according to the distance from threats and the growth trend of the tree. It solves the problem that some dynamic step size RRTs cause increasing the path length and slowing down the searching efficiency. In addition, a path length constraint strategy is designed to satisfy the fuel-saving requirement, and it is implemented in the path searching process. This strategy tests the potential tree node and accepts it if the estimated path length less than or equal to the maximum allowed value of path length.

2. The basic RRT

The pseudocode of the basic RRT is presented in Algorithm 1, where \( T \) denotes the tree and \( C \) is the configuration space. The tree is initialized by setting the start node \( q_{\text{start}} \) as the root, then the RRT incrementally builds the tree, with a fixed step length \( \varepsilon \), by adding a new node \( q_{\text{new}} \) towards the randomly selected node \( q_{\text{rand}} \).

The algorithm relies on functions to execute single operations. Function RandomState is used to randomly choose a node \( q_{\text{rand}} \) in \( C \), and the randomly selected node is going to guide the expansion direction of the tree. Function NearestNeighbor finds the nearest node to \( q_{\text{rand}} \) in the tree as \( q_{\text{near}} \). Then function NewState calculates the new node \( q_{\text{new}} \) in the direction from \( q_{\text{near}} \) to \( q_{\text{rand}} \) with a certain step size \( \varepsilon \), as shown in Figure 1. If the new node \( q_{\text{new}} \) and the edge between \( q_{\text{near}} \) and \( q_{\text{new}} \) are in the free space of \( C \), then function ExtendTree adds them to the tree. When the distance between \( q_{\text{new}} \) and \( q_{\text{goal}} \) is less than a preset threshold value \( d_{\text{min}} \), RRT returns the tree and path. The RRT runs for a preset number of iterations or a fixed amount of time, and returns failure if time runs out.

| Algorithm 1: Basic RRT |
|------------------------|
| 1 \( T\text{.init}(q_{\text{init}}); \) |
| 2 \( \text{for } i=1 \text{ to } M \text{ do} \) |
| 3 \( q_{\text{rand}} \leftarrow \text{RandomState}(C); \) |
| 4 \( q_{\text{near}} \leftarrow \text{NearestNeighbor}(T, q_{\text{rand}}); \) |
| 5 \( q_{\text{new}} \leftarrow \text{NewState}(q_{\text{near}}, \varepsilon, q_{\text{rand}}); \) |
| 6 \( \text{if CollisionFree}(q_{\text{near}}, q_{\text{new}}) \text{ then} \) |
| 7 \( \text{ExtendTree}(T, q_{\text{new}}); \) |
| 8 \( \text{if } |q_{\text{new}} - q_{\text{goal}}| \leq d_{\text{min}} \text{ then} \) |
| 9 \( \text{Return } T \& \text{ Path} \) |
| 10 \( \text{Return Failure} \) |
3. Description of planning mission and constraints
This section mainly explains the UAV path planning mission and introduces related constraints that should be taken into account when dealing with the UAV path planning mission.

3.1. Description of the UAV path planning mission
When the UAV follows the original route and detects some obstacles or threats that have not been detected before, it needs to plan a local path quickly to avoid them, and then return to the original route to continue the flight mission. Let $S$ be the start node of the re-plan path in a local area, and $G$ refers to the end node. For example, Figure 2 shows the original UAV path in the local area and the sudden appearing threats are represented by black circular objects. These threats make the original path no longer meet the requirement of safe flight, and the planner must re-plan a feasible path rapidly to avoid these threats. In addition, the planner should have good adaptability in dealing with complex environments as threats can be a dense cluster of obstacles that is hard to get around.

3.2. Description of constraints
As basic RRT searches for solutions completely random, usually the solution path generated by the basic RRT cannot satisfy the flight constraint requirement of the UAV. To overcome this shortcoming, several related constraints explained in Literatures [10, 22] should be supplemented into RRT algorithm when using RRT and its variants to solve UAV path planning problems.

3.2.1. Minimum flight length constraint.
Minimum flight length constraint limits the shortest distance that the UAV should keep flight forward before changing its posture or direction, expressing as $l_i \geq l_{\text{min}}, i = 1, \cdots, m$. Here $l_{\text{min}}$ denotes the minimum distance and $l_i$ represents the length of ith UAV path segment. In RRT and its variants, this constraint is going to restrict the step size of the tree,
expressing as $\varepsilon_j \geq \varepsilon_{\text{min}}$. Here $\varepsilon_j$ is the step size in current iteration and $\varepsilon_{\text{min}}$ is the minimum flight length, which should be equal to $l_{\text{min}}$.

3.2.2. Maximum flight length constraint. Since the maximum flight length is limited by the amount of fuel and flight time, the path length of the solution must be no more than a preset maximum flight length. Let $L_{\text{max}}$ indicates the maximum flight length, then the maximum flight length constraint can be expressed by Equation (1).

$$L = \sum_{i=1}^{m} l_i, \quad L \leq L_{\text{max}}$$

3.2.3. Expansion direction constraint. When RRT grows, the randomly generated node $q_{\text{rand}}$ provides the expansion direction for it. Assuming that $\theta$ is the angle between the expansion direction and the current flight direction (as shown in Figure 3). If $\theta$ is greater than the max angle, which is restricted by UAV performance and denoted as $\theta_{\text{max}}$, the generated path cannot be tracked well by the UAV. Therefore, expansion direction of the tree should be restrained, and this constraint can be expressed as $\theta \leq \theta_{\text{max}}$.

In this study, it is assumed that the flight altitude is constant, and the flying elevation angle and subduction angle are not considered.

4. The proposed method and the basic strategies

This section is the core part of the study, it presents an improved dynamic step size strategy and a path length constraint strategy, and then proposes the improved RRT algorithm.

4.1. The improved dynamic step size strategy

In the traditional RRT algorithm, the step size is constant during the searching process for a feasible path. When the planner is working in an area with dense obstacles, choosing a large step size is not reasonable to improve the success rate of tree expansion. While in an area with sparse obstacles, choosing a small step size is not reasonable to improve the computational time. Literatures [20-22] proposed to dynamically select the step size during the tree growth. Inspired by that, an improved dynamic step size strategy is proposed, in which the step size is determined after a comprehensive consideration of the distance from obstacles and the growth trend of the tree. The improved dynamic step size strategy generates a step size $\varepsilon_j$ by using the Equation (2), in which, $\varepsilon$ is the original step size, $k$ is the minimum step size coefficient and $\varepsilon_{\text{min}}$ is the minimum flight length.

$$\varepsilon_j = \begin{cases} \varepsilon, & d > d_r \text{ and } \varphi \leq \varphi_{\text{max}} \\ k \cdot \varepsilon_{\text{min}}, & 1 \leq k < \varepsilon \cdot \varepsilon_{\text{min}}^{-1}, \text{otherwise} \end{cases}$$

As shown in Figure 4, $d_r$ is a preset safe distance value and $d$ refers to the distance between the node $q_{\text{near}}$ and the nearest threat in current iteration. If the node $q_{\text{near}}$ is located far from the threat, it is safe to set a large step size. Conversely, if $q_{\text{near}}$ is close to the threat, a small step size should be taken to avoid collision. However, if the step size is chosen only based on the distance from threats, without considering the expansion direction of the tree, the tree may grow toward an undesirable direction at a large step size. As a result, this will slow down the searching efficiency. Additionally, the solution path will be far from optimal.

As shown in Figure 5, $\varphi$ represents the angle between the expansion direction and the target direction, and $\varphi_{\text{max}}$ is a preset threshold value. It is considered that the tree is growing toward the
target region if angle $\varphi$ satisfies the condition of $\varphi \leq \varphi_{\text{max}}$ in the current iteration, and it is highly efficient to obtain a solution by expanding the tree toward the target region with large steps.

![Figure 4](image1.png)  
**Figure 4.** Distance $d$ and safe distance $d_s$.

![Figure 5](image2.png)  
**Figure 5.** The angle $\varphi$ between the expansion direction and the target direction.

Therefore, the dynamic step size strategy proposed in this work provides step sizes according to the distance from obstacles and the growth trend of the tree. The main idea of the improved dynamic step size strategy is that if $q_{\text{near}}$ locates far from obstacles ($d > d_s$) and the tree is expanding toward the target area ($\varphi \leq \varphi_{\text{max}}$) in the current iteration, the current step size $\varepsilon_f$ can take the large value $\varepsilon$. Otherwise, $\varepsilon_f$ should be the smaller value $k_{\text{min}} \cdot \varepsilon_{\text{min}}$.

### 4.2. Path length constraint strategy

In this work, responding quickly and avoiding obstacles effectively are considered the first and foremost. Instead of searching for the most optimal path, the path length constraint strategy is used to measure and limit the path length to guarantee that it is within the maximum path length allotted. Once a new node $q_{\text{new}}$ is obtained, it should be tested whether the new node $q_{\text{new}}$ satisfies the requirement of maximum path length constraint. Since the path is not completely obtained during the tree growing, the actual length of the path is unobtainable by using Equation (1). Inspired by A* algorithm [1-3], the path length can be divided into two parts, as shown in Equation (3). One is the completed path length represented as $L_1$, which can be obtained by the superposition calculation of step sizes from the start node $q_{\text{start}}$ to the new node $q_{\text{new}}$, as shown in Figure 6. The other is the length of the unplanned part denoted as $L_2$, and it can be estimated based on the linear distance between $q_{\text{new}}$ and $q_{\text{goal}}$.

$$L = L_1 + L_2, \quad L_1 = \sum_{i=1}^{m} \varepsilon_f^i, \quad L_2 = d_{\text{ng}}$$

![Figure 6](image3.png)  
**Figure 6.** Path diagram of the UAV.

Since the actual path is usually longer than the straight-line path between $q_{\text{new}}$ and $q_{\text{goal}}$, especially in a complex environment, a coefficient $S$ is introduced to properly enlarge the estimated part of the path length. Finally, in this work, the path length $L$ is computed and constrained according to Equation (4). Here $L_{\text{max}}$ refers to the maximum permitting path length of the UAV.

$$L = \sum_{i=1}^{m} \varepsilon_f^i + S \cdot d_{\text{ng}}, \quad L \leq L_{\text{max}}$$
Coefficient $S$ can be a constant or a variable value, and it should never be less than 1 as $L$ never less than the value of the linear distance between $q_{\text{new}}$ and $q_{\text{goal}}$. The value of $S$ affects the performance of the proposed algorithm in aspects of path length and running time. If $S$ is a small value that close to 1, then the path length constraint represented in Equation (4) performs relax in restricting path length, and a new node is easy to satisfy the requirement of $L \leq L_{\text{max}}$. As a result, the process of generating a feasible path is efficient, but the generated path is sub-optimal. As the value of $S$ increases, the constraint becomes stronger, the solution path tends to be the optimal one with the shortest path length, but it will consume more time in searching for feasible new nodes.

**Figure 7.** Process of the proposed algorithm.

4.3. Process of the proposed algorithm

The process of the proposed method is exhibited in Figure 7. Compared with the basic RRT algorithm, there are four different procedures in the proposed algorithm, and they are explained as follows.

1) **Biased-goal sampling strategy** is a sampling method used to obtain $q_{\text{rand}}$ in the biased RRT algorithm [16]. It selects the target node $q_{\text{goal}}$ with a certain probability $p$ and selects random node in configuration space $C$ with probability $1-p$. This strategy is used to bias the growth of the tree towards the target node, and it is introduced in the proposed algorithm to speed up the searching process for feasible solutions.

2) **Expansion direction constraint strategy** is used to determine whether the UAV can realize the change of the current flight direction. If the node $q_{\text{rand}}$ provides an infeasible direction for the UAV, this strategy instantly abandons the current $q_{\text{rand}}$ and ends the current iteration.

3) **Dynamic step size strategy** proposed in Section 4.1 is applied to select step size in the current iteration. And the step size will be used to calculate the new node $q_{\text{new}}$.

4) **Path length constraint strategy** provided in Section 4.2 is used to estimate the path length $L$. The new node $q_{\text{new}}$ is abandoned if it makes the path does not satisfy the requirement of $L \leq L_{\text{max}}$, and then the algorithm turns to perform the next iteration immediately.
In this work, the principle for arranging the sequence of these procedures is that ends the current iteration as quickly as possible if relevant constraints are not to be satisfied. Then performs the next iteration instantly to find a feasible potential new node. This principle is beneficial for saving computational time. For example, the expansion direction constraint strategy is arranged to perform after a randomly selected node $q_{\text{rand}}$ is obtained and the nearest node $q_{\text{near}}$ is found, but not after the new node $q_{\text{new}}$ is calculated. The reason is that nodes $q_{\text{rand}}$ and $q_{\text{new}}$ provide the same information of expansion direction for the tree but it consumes more time to get the new node $q_{\text{new}}$.

5. Simulation results

5.1. Simulation design

In this section, two simulation experiments are carried out to investigate the performance of the proposed algorithm in a complex and trapped environment. In both experiments, the running time and the path length are the focus of attention. The first experiment is to investigate how the coefficient $S$ in the path length constraint strategy affects the performance of the proposed algorithm. The second test is to compare the performance of the proposed algorithm with other two dynamic step size RRT methods in Literatures [20, 21].

The simulation environment map is 500m by 500m, and obstacles and threats are simulated as circular objects. UAV path planning begins at the initial location of [10, 10] and ends at position [460, 460]. The relevant parameters should be adjusted according to the environment. In this simulation work, parameters of the proposed algorithm are presented in Table 1. To guarantee a fair comparison, all simulation experiments are conducted on the same computer.

| Parameters | Value |
|------------|-------|
| $\theta_{\text{max}}/^{\circ}$ | 60 |
| $\phi_{\text{max}}/^{\circ}$ | 30 |
| $p$ | 0.1 |
| $\varepsilon_{\min}$ | 10 |
| $\varepsilon$ | 30 |
| $d_s$ | 30 |
| $L_{\text{max}}$ | 1020 |
| $k$ | 1 |

5.2. Results

The aim of the first experiment is to investigate the proposed algorithms’ performance with different $S$, and Figure 8 shows the experimental results. As $S$ increases from 1 to 1.25, the path length and the running time are changed very little. The reason for this phenomenon is that the path length constraint strategy performs relax in restricting path length when the value of $S$ is selected from 1 to 1.25, and it is easy for the new node to pass the path length constraint test. However, as the $S$ parameter continues to grow from 1.25 to 1.45, the constraint is strengthened, and the algorithm tends to search for optimal solutions with the shortest path length. Therefore, when $S$ increases in the interval of 1.25 to 1.45, the path length significantly reduced, and the running time increases rapidly as the Figure 8 shows.

**Figure 8.** Performance of the proposed algorithm with different $S$.

It is beneficial to set a large value of $S$ when dealing with problems of finding the shortest path. Since the objective of this work is to reduce the planning time for the UAV path planning, the
parameter $S$ should not be a value larger than 1.25. Figure 9(a) shows the simulation environment, Figure 9(b) and Figure 9(c) show the paths of the proposed algorithm with $S=1.2$ to bypass the trapped area from different sides.

![Simulation Environment](image1)

![The Proposed Algorithm](image2)

![The Proposed Algorithm](image3)

**Figure 9.** Paths of the proposed algorithm with $S=1.2$.

The second simulation experiment is to compare the performances among the proposed algorithm and other two dynamic step size algorithms. Coefficient $S$ is set at $S=1.2$ for the proposed algorithm in this experiment. Table 2 presents the comparison results of the proposed algorithm and algorithms in Literatures [20, 21], and the comparison results are measured in terms of the path length, the running time, the number of tree nodes, and the number of iterations until convergence. Results in Table 2 are evaluated by independently performing 100 times.

The algorithm in Literature [20] chooses the step size depending on $h$, the number of levels down the tree from the root node. The function to calculate the step size is $\varepsilon_j = \varepsilon_0 \sqrt{1 + h}$, and here $\varepsilon_0$ refers to the initial step size. The performances of the algorithm [20] with three different initial step sizes are listed in the Table 2, and the best result is obtained when the initial step is $\varepsilon_0 = 2$.

**Table 2.** Performance comparison of the three algorithms.

| Algorithm | Path length | Run time/s | Tree nodes | Avg. Number of Iterations |
|-----------|-------------|------------|------------|---------------------------|
| 1 The proposed algorithm | 912 | 42 | 0.09 | 0.08 | 178 | 56 | 575 |
| 2 Algorithm in [20] ($\varepsilon_0 = 2$) | 1022 | 62 | 1.28 | 5.41 | 203 | 130 | 401 |
| 3 Algorithm in [20] ($\varepsilon_0 = 5$) | 1059 | 93 | 2.34 | 5.81 | 619 | 518 | 1287 |
| 4 Algorithm in [20] ($\varepsilon_0 = 10$) | 1113 | 127 | 3.86 | 9.52 | 702 | 593 | 1798 |
| 5 Algorithm in [21] without optimization | 1152 | 143 | 0.09 | 0.13 | 177 | 113 | 348 |
| 6 Algorithm in [21] with optimization | 1040 | 133 | 0.12 | 0.09 | 177 | 113 | 376 |

The method in Literature [21] separates the planning process into two parts, establishing a candidate path (without the optimization process) and optimizing the path. The performances of the algorithm [21] with the optimization process and the algorithm without the optimization process are exhibited in Table 2.

As shown in Table 2, the path length obtained in the proposed algorithm is less than the value of $L_{\text{max}}$, and it is much shorter than the others. The running time of the proposed method to search for a solution path is 0.09s, and it is much shorter than the running time of the algorithm in [20]. Although the running time of the algorithm in [21] without optimization is equal to that of the proposed algorithm, the path length is much longer than that of the proposed method. That is to say, the
algorithm in this work has a good comprehensive performance in terms of the running time and the path length.

The number of tree nodes represents the number of successful expansions of the tree, which to a certain extent reflects the algorithm’s exploration scale before finding a solution. In Table 2, the number of tree nodes of the proposed algorithm is 178, it is almost the same as that obtained by the algorithm in [21], but it is less than the value of the algorithm in [20]. This indicates that compared with the algorithm in [20], the proposed algorithm can find a solution within a less exploration time. And this is consistent with the comparison results of the algorithms in terms of the running time.

It is notable that compared with the algorithm [20] (\(\epsilon_0 = 2\)) and the algorithm [21], the proposed algorithm performs the largest number of iterations. However, it does not take the longest running time. The reason for this phenomenon is that the proposed algorithm ends the current iteration and executes the next iteration immediately, once one of the constraints is not satisfied (as shown in Figure 7). Although the proposed algorithm results in an increase in the number of iterations, it is more efficient than ending the current iteration after a collision occurs.

6. Conclusions and future work
In this paper, an improved RRT algorithm for solving UAV path planning problems in complex environments is proposed. A new dynamic step size strategy is introduced, and it adjusts the step size according to the distance from the obstacle and the growth direction of the tree. A path length constraint strategy is designed to limit the path length during the solution searching process. The simulation experiments demonstrate that the proposed algorithm performs well in decreasing the running time and reducing the path length. Here obstacles are considered stationary during the searching process for a solution, and the proposed method can now be implemented offline.

To realize the online path planning of UAVs, more kinodynamic constraints of UAV would be introduced, such as speed constraint. And more factors should be considered, such as real-time locations and status of the UAV and obstacles. How to adjust the proposed algorithm and use it to solve the UAV path planning problems in a 3D environment with moving obstacles? How to adaptively change the \(S\) value according to both the planning objective and the distribution of obstacles in the environment? In the future work, these problems need to be studied further.

Acknowledgements
The paper is supported by the Research Project of Changzhou College of Information Technology (CXKZ201904Y), the Enterprise Practical Training Project for Young Teachers of Higher Vocational Colleges in Jiangsu Province (2020QYSJ130), and the Changzhou Key Laboratory of High Technology (CM20183004).

References
[1] Hart P E, Nilsson N J and Raphael B 1968 A formal basis for the heuristic determination *IEEE Transactions of Systems Science and Cybernetics* ssc-4 100-7
[2] Erke S, Bin D, Yiming N, Qi Z, Liang X and Dawei Z 2020 An improved a-star based path planning algorithm for autonomous land vehicles *International Journal of Advanced Robotic Systems* 17 1-13
[3] Guruji K, Agarwal H and Parsediya K 2016 Time-efficient a* algorithm for robot path planning *Procedia Technology* 23 144-9
[4] Hussein Qahtan A G and Ali Emad 2020 A Path-planning dynamic 3d space using modified a* algorithm *IOP Conference Series: Materials Science and Engineering* 928 1-13
[5] Chen Y, Pi D and Xu Y 2021 Neighborhood global learning based flower pollination algorithm and its application to unmanned aerial vehicle path planning *Expert Syst. Appl.* 170 1-16
[6] Roberge V, Tarbouchi M and Labonte G 2018 Fast genetic algorithm path planner for fixed-wing military uav using gpu *IEEE T. Aero. Elec. Sys.* 54 2105-17
[7] Ajeil F H, Ibraheem I K, Sahib M A and Humaidi A J 2020 Multi-objective path planning of an autonomous mobile robot using hybrid pso-mfb optimization algorithm *Appl. Soft Comput.* **89** 1-28

[8] Hamad I I and Hasan M S 2020 A review: on using aco based hybrid algorithms for path planning of multi-mobile robotics *International Journal of Interactive Mobile Technologies* **14** 145-56

[9] Zhao Y, Zheng Z and Liu Y 2018 Survey on computational-intelligence-based UAV path planning *Knowl.-Based Syst.* **158** 54-64

[10] Li B, Qi X, Yu B and Liu L 2020 Trajectory planning for uav based on improved aco algorithm *IEEE Access* **8** 2995-3006

[11] Tran N H, Nguyen A D and Nguyen T N 2020 A genetic algorithm application in planning path using b-spline model for autonomous underwater vehicle (AUV) *Applied Mechanics and Materials* **902** 54-64

[12] Krell E, Sheta A, Balasubramanian A P R and King S A 2019 Collision-free autonomous robot navigation in unknown environments utilizing pso for path planning *Journal of Artificial Intelligence and Soft Computing Research* **9** 267-82

[13] Lavalle S M 1998 Rapidly-exploring random trees: a new tool for path planning (Computer Science Department, Iowa State University) Technical Report 98-11

[14] Shu X, Ni F, Zhou Z, Liu Y, Liu H and Zou T 2019 Locally guided multiple bi-rrt for fast path planning in narrow passages *Proc. of the IEEE Int. Conf. on Robotics and Biomimetics* vol 1 (Piscataway, N.J.: IEEE) pp 2085-91

[15] Luo R C and Kuo C 2016 Intelligent seven-dof robot with dynamic obstacle avoidance and 3-d object recognition for industrial cyber–physical systems in manufacturing automation *P. IEEE* **104** 1102-13

[16] Yuan C, Liu G, Zhang W and Pan X 2020 An efficient rrt cache method in dynamic environments for path planning *Robot. Auton. Syst.* **131** 1-12

[17] Kuffner J J and Lavalle S M 2000 Rrt-connect: an efficient approach to single-query path planning *IEEE Int. Conf. on Robotics and Automation.* (San Francisco) pp 995-1001

[18] Otte M and Correll N 2011 Path planning with forests of random trees: parallelization with super linear speedup (Department of Computer Science, University of Colorado at Boulder) Technical Report CU-CS 1079-11

[19] Wang W, Zuo L and Xu X 2018 A learning-based multi-rrt approach for robot path planning in narrow passages *J. Intell. Robot. Syst.* **90** 81-100

[20] Mccourt M, Ton C T, Mehta S S and Curtis J W 2016 Adaptive step-length rrt algorithm for improved coverage *AIAA Guidance, Navigation, and Control Conf.* (San Diego, California) vol 1 pp 1-8

[21] Wang C and Meng M Q 2016 Variant step size rrt: an efficient path planner for uav in complex environments *2016 IEEE Int. Conf. on Real-time Computing and Robotics* (Angkor Wat, Cambodia) pp 555-60

[22] Cheng H H, Yang S and Qi X H 2018 Dynamic trajectory planing of quadrotors uav based on improved rrt algorithm *Computer Engineering and Design* **39** 3705-11