

Article

Dubins Path-Oriented Rapidly Exploring Random Tree* for Three-Dimensional Path Planning of Unmanned Aerial Vehicles

Youyoung Yang ^{1,†} , Henzeh Leeghim ^{1,†} and Donghoon Kim ^{2,*} 

¹ Department of Aerospace Engineering, Chosun University, Gwangju 61452, Korea; youyoung.yang@controla.re.kr (Y.Y.); h.leeghim@controla.re.kr (H.L.)

² Department of Aerospace Engineering & Engineering Mechanics, University of Cincinnati, Cincinnati, OH 45221, USA

* Correspondence: donghoon.kim@uc.edu

† These authors contributed equally to this work.

Abstract: Unmanned aerial vehicles (UAVs) do not collide with obstacles, generate a path in real-time, and must fly to the target point. The sampling-based rapidly exploring random tree (RRT) algorithm has the advantages of fast computation and low computational complexity. It is suitable for real-time path generation, but the optimal path cannot be guaranteed. Further, the direction of the flight and the minimum radius of rotation have not been taken into account for the characteristics of the UAVs. This work proposes a Dubins path-oriented RRT* algorithm, which applies the Dubins path to the RRT algorithm to consider the direction of flight and the minimum radius of rotation and improves optimality and convergence. The proposed algorithm sets the sample node as the target point, orients toward the Dubins path, and then generates a tree. To verify the performance of the proposed algorithm, it is compared with existing RRT algorithms. As a result of performance analysis, the proposed algorithm improved the path length by 14.87% and the calculation time by 82.36%. Finally, the algorithm's performance is verified by applying the proposed algorithm to a fixed-wing UAV and performing a numerical analysis of the generated path.

Keywords: path planning; rapidly-exploring random tree; Dubins path; unmanned aerial vehicle



Citation: Yang, Y.; Leeghim, H.; Kim, D. Dubins Path-Oriented Rapidly Exploring Random Tree* for Three-Dimensional Path Planning of Unmanned Aerial Vehicles.

Electronics **2022**, *11*, 2338. <https://doi.org/10.3390/electronics11152338>

Academic Editor: Ahmad Taher Azar

Received: 10 June 2022

Accepted: 22 July 2022

Published: 27 July 2022

Publisher's Note: MDPI stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.



Copyright: © 2022 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

1. Introduction

With technological advancements, unmanned aerial vehicles (UAVs) are being used in various industries and recreational fields, with notable applications in disaster assessment, diagnosis, measurement, relief, etc. [1–3]. Ensuring safety and reliability should be prioritized to carry out smooth missions. That is, it is necessary to generate the shortest path to the destination without colliding with obstacles [4,5]. Typical path generation algorithms include grid-based, mathematical model-based, and sampling-based approaches [6].

Grid-based algorithms, such as Dijkstra, A*, and D*, are intuitive and search by dividing all sections evenly, reducing the probability of falling into the local maximum [7–9]. However, there is a disadvantage in that the calculation time increases exponentially depending on the search section [10]. Mathematical model-based algorithms, such as linear algorithms and optimal control, use constraints to achieve optimal solutions by modeling given surroundings and dynamic systems [6,11]. Although an optimal path can be generated through these methods, it takes a lot of time to model environmental conditions and calculate constraints. Sampling-based algorithms retrieve a node adjacent to a randomly selected point in a feasible space and expand the node if possible. This is repeated until the path to the destination is generated. The main advantages of sampling-based algorithms are low computational costs and applicability to high-dimensional problems. Typical sampling-based algorithms include probabilistic roadmaps (PRM) and rapidly exploring

random tree (RRT). PRM is mainly used in fixed obstacle environments [12], while RRT is suitable for generating paths through tree extensions and also for use in environmental changes [10].

RRT has the advantage of quickly generating a path with a light computation burden but has limitations in that the generated path is not smooth and cannot guarantee an optimal path. RRT* holds the fundamentals of the RRT but guarantees gradual convergence to the optimal path through additional steps [13]. However, RRT* is not suitable for UAVs that operate in real-time due to expensive computational costs. An improved biased RRT is developed to aim at the target point with a certain probability in RRT [14–16]. Convergence increases by oriented toward a target point, but the convergence becomes poor when there are many obstacles.

This work applies the Dubins path to the RRT* to improve the convergence of the RRT for the path generation of fixed-wing UAVs and reduce the calculation time. The authors considered the flight direction and the minimum turning radius, which have not been considered in the RRT. Dubins path uses the Dubins curve proposed by Dubins to generate the shortest path considering the rotation radius limited to the position and direction at the given initial and final points [17]. As a result, the proposed algorithm reduces the cost of path generation computation and avoids obstacles by limiting unnecessary tree extensions. In addition, it aims at Dubins path to ensure optimality and considers the flight direction and the minimum turning radius. The simulation study is performed with a fixed-wing UAV model to confirm the characteristics of the algorithm. To verify the performance of the proposed algorithm, the simulation results are compared to RRT, RRT*, and biased RRT.

2. Previous Research

RRT, one of the sampling-based algorithms, is a method of finding a path by randomly generating several sample nodes without dividing a given space into a grid. It efficiently identifies the barrier-free space by checking whether the sample node or the line connecting the two-sample nodes collides with the obstacle. The sampling-based algorithm can be used in a high-dimensional space and has the advantage of being able to quickly generate a path due to a small amount of calculation. The basic idea is to build a tree that makes up nodes and connections. In the tree structure, all nodes are connected to one other node called a parent, and the starting point is the top parent of the tree. RRT calculates a tree connecting the starting point and the target point in a given space.

2.1. RRT Algorithm

RRT was proposed by LaValle [18], and Algorithm 1 and Figure 1 depict the pseudocode and the path generation process of the RRT, respectively.

Algorithm 1 RRT

```

1: procedure RRT( $q_s, q_f$ )
2:    $T \leftarrow \text{InitialSetting}()$ 
3:   while  $\text{norm}(q_f - q_{new}) \geq R_{\text{threshold}}$  do
4:      $q_{rand} \leftarrow \text{RandomSamling}()$ 
5:      $q_{nearest} \leftarrow \text{FindNearest}(T, q_{rand})$ 
6:      $q_{new} \leftarrow \text{Extend}(q_{nearest}, q_{rand}, \epsilon)$ 
7:     if  $\text{CollisionCheck}(q_{nearest}, q_{new})$  then
8:        $T \leftarrow q_{new}$ 
9:     end if
10:  end while
11:   $N_{path}, t_{cal} \leftarrow \text{SelectPath}(T, q_f)$ 
12:  return  $N_{path}, t_{cal}$ 
13: end procedure

```

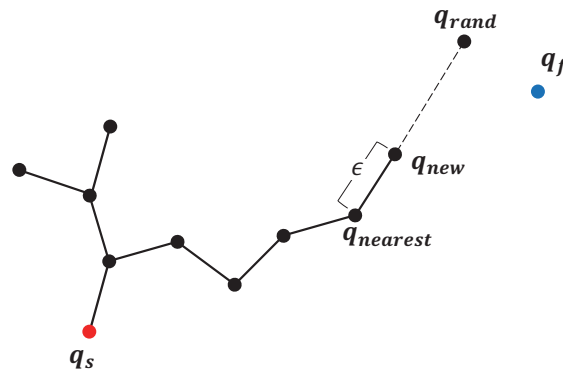


Figure 1. Illustration of the RRT process.

RRT aims to extend the tree through an obstacle check between the tree T starting from the start point q_s and the randomly generated node q_{rand} in a given space, generating a path to reach the target point q_f . A sample node q_{rand} is generated, the distance between q_{rand} and all nodes constituting the tree is measured, and the nearest node $q_{nearest}$ is set. Then, it creates q_{new} at a point ϵ distance away from $q_{nearest}$ in the q_{rand} direction. When q_{new} is created, an obstacle check is performed between q_{new} and $q_{nearest}$. If there is no obstacle between the two nodes, q_{new} is added to expand the tree. If there is an obstacle, the process of creating q_{rand} is repeated. It repeats until a certain number of nodes are satisfied, such as generating q_{rand} or q_{new} or until the tree reaches a certain radius of q_f . When the tree is expanded, it connects q_f and the tree and starts with q_f and performs the process of finding the parent of the previously created node. This is repeated until q_s is reached to generate a path.

2.2. RRT* Algorithm

RRT* is an algorithm that improves optimality by adding a process of re-selecting parent nodes and re-connecting trees to the RRT [19]. The pseudocode and the path generation process of the RRT* are shown in Algorithm 2 and Figure 2, respectively.

Algorithm 2 RRT*

```

1: procedure RRT*( $q_s, q_f$ )
2:    $T \leftarrow \text{InitialSetting}()$ 
3:   while  $\text{norm}(q_f - q_{new}) \geq R_{\text{threshold}}$  do
4:      $q_{rand} \leftarrow \text{RandomSamling}()$ 
5:      $q_{nearest} \leftarrow \text{FindNearest}(T, q_{rand})$ 
6:      $q_{new} \leftarrow \text{Extend}(q_{nearest}, q_{rand}, \epsilon)$ 
7:     if  $\text{CollisionCheck}(q_{nearest}, q_{new})$  then
8:        $T \leftarrow q_{new}$ 
9:        $Q_{\text{near}}, q_{\text{min}} \leftarrow \text{BestParent}(T, q_{new}, R^*)$ 
10:       $T \leftarrow \text{Rewiring}(T, Q_{\text{near}}, q_{new}, q_{\text{min}}, R^*)$ 
11:     end if
12:   end while
13:    $N_{\text{path}}, t_{\text{cal}} \leftarrow \text{SelectPath}(T, q_f)$ 
14:   return  $N_{\text{path}}, t_{\text{cal}}$ 
15: end procedure

```

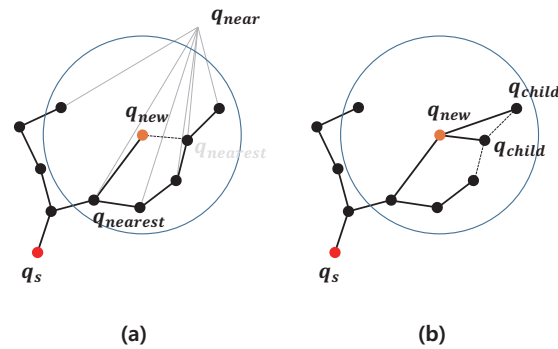


Figure 2. Illustration of the RRT* process: (a) Best parent algorithm. (b) Rewiring algorithm.

RRT* finds the node closest to q_{rand} , sets it to $q_{nearest}$, and creates q_{new} at points ϵ distance away in the q_{rand} direction. When q_{new} is created, as shown in Figure 2a, nodes that are within a certain radius around q_{new} and do not get hit by obstacles are set as q_{near} , and the node with the minimum path cost to q_{new} among q_{near} is found. If it finds a q_{near} with a lower path cost than the previously connected $q_{nearest}$, it disconnects from $q_{nearest}$, sets the node with low path cost as the parent node q_{near} of q_{new} , and re-connects the tree. The path cost is calculated by using q_{new} as a parent node for q_{near} within a certain radius around q_{new} . If the path cost is lower when connected through q_{new} than the existing path cost of q_{near} , the tree that was previously connected to q_{near} disconnects and re-connects q_{new} and the tree, as shown in Figure 2b. RRT* complements optimality, which is a disadvantage of the RRT, through the best parent process of re-connecting $q_{nearest}$ and the re-wiring process of reducing the path cost of q_{near} .

2.3. Biased RRT Algorithm

The biased RRT is an improved algorithm that supplements the node generation method in the RRT and improves the randomly sampled nodes to a target point with a certain probability [14]. If the probability of aiming for the target point is high, convergence is increased in the process of expanding the tree. However, the map cannot be searched as a whole, so it takes more time to get out of the obstacle than before. There is a deviation in the path generation time depending on the complexity and directionality probability of the obstacle. Because the operating environment is not the same every time, it is generally known that using a directivity probability of 10% is the most effective regardless of the density of obstacles [9]. The pseudocode and the path generation process of the biased RRT are shown in Algorithm 3 and Figure 3, respectively.

Algorithm 3 Biased RRT

```

1: procedure BRRT( $q_s, q_f$ )
2:    $T \leftarrow \text{InitialSetting}()$ 
3:   while  $\text{norm}(q_f - q_{new}) \geq R_{\text{threshold}}$  do
4:      $q_{rand} \leftarrow \text{RandomSampling}(k, q_f)$ 
5:      $q_{nearest} \leftarrow \text{FindNearest}(T, q_{rand})$ 
6:      $q_{new} \leftarrow \text{Extend}(q_{nearest}, q_{rand}, \epsilon)$ 
7:     if  $\text{CollisionCheck}(q_{nearest}, q_{new})$  then
8:        $T \leftarrow q_{new}$ 
9:     end if
10:  end while
11:   $N_{\text{path}}, t_{\text{cal}} \leftarrow \text{SelectPath}(T, q_f)$ 
12:  return  $N_{\text{path}}, t_{\text{cal}}$ 
13: end procedure

```

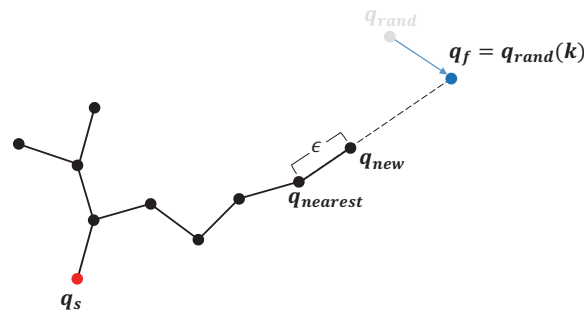


Figure 3. Illustration of the biased RRT process.

In the biased RRT, only the process of generating q_{rand} from the RRT has been changed, and the rest of the process is the same as in the RRT. In the RRT, it extends the tree toward q_f by replacing q_{rand} , which is randomly sampled, with q_f according to a certain probability. The directivity probability that directs q_f to a certain probability is called k , and the tree expands toward q_f for each k probability so that it has convergence. When q'_{new} generated by q_f collides with an obstacle to solve these problems, q_{rand} is used to generate q_{new} in the direction without obstacles and extend the tree.

3. Dubins Path-Oriented RRT* Algorithm

The pseudocode of the Dubins path-oriented RRT* proposed is shown in Algorithm 4, and the path generation process of the algorithm is shown in Figures 4 and 5. The Dubins path-oriented RRT* complements the optimality and convergence of the RRT and utilizes the Dubins path to consider the flight direction and minimum radius of rotation of the UAV. The proposed algorithm has two methods for complementing optimality. The first method generates the Dubins path in the position and direction of the initial and final points. It is oriented toward the generated Dubins path and generates a path. The second method, similar to the RRT*, performs the process of re-selecting the parent node and re-constructing the tree. The method for compensating convergence has a 100% probability of replacing q_{rand} with q_f , so convergence to the target is further obtained than in the biased RRT. However, when the percentage of the obstacle space is increased, q_{new} is not generated properly like the biased RRT, resulting in the result of being blocked by obstacles and reduced convergence.

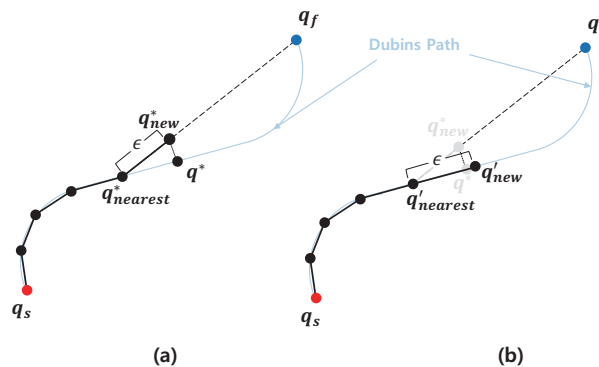


Figure 4. Optimization improvement method: (a) Closest point search algorithm. (b) Extend algorithm.

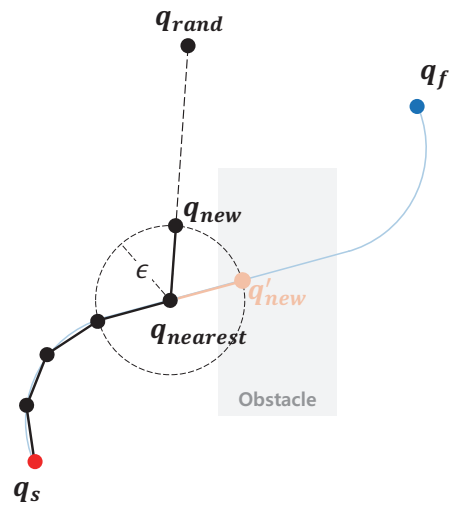


Figure 5. Convergence improvement method.

Algorithm 4 DRRT*

```

1: procedure DRRT*( $q_s, q_f, \text{dubins path}$ )
2:    $\mathbf{T} \leftarrow \text{InitialSetting}()$ 
3:   while  $\text{norm}(q_f - q_{new}) \geq R_{\text{threshold}}$  do
4:      $q_{rand} \leftarrow \text{RandomSamling}()$ 
5:      $q_{nearest}^* \leftarrow \text{FindNearest}(\mathbf{T}, q_f)$ 
6:      $q_{new}^* \leftarrow \text{Extend}(q_{nearest}^*, q_f, \epsilon)$ 
7:      $q^* \leftarrow \text{ColsestPoint}(q_{new}^*, q_s, q_f, \text{dubins path})$ 
8:      $q'_{nearest} \leftarrow \text{FindNearest}(\mathbf{T}, q^*)$ 
9:      $q'_{new} \leftarrow \text{Extend}(q'_{nearest}, q^*, \epsilon)$ 
10:    if  $\text{CollisionCheck}(q'_{nearest}, q'_{new})$  then
11:       $\mathbf{T} \leftarrow q'_{new}$ 
12:       $\mathbf{Q}_{near}, q_{min} \leftarrow \text{BestParent}(\mathbf{T}, q'_{new}, R^*)$ 
13:       $\mathbf{T} \leftarrow \text{Rewiring}(\mathbf{T}, \mathbf{Q}_{near}, q'_{new}, q_{min}, R^*)$ 
14:    else
15:       $q_{nearest} \leftarrow \text{FindNearest}(\mathbf{T}, q_{rand})$ 
16:       $q_{new} \leftarrow \text{Extend}(q_{nearest}, q_{rand}, \epsilon)$ 
17:      if  $\text{CollisionCheck}(q_{nearest}, q_{new})$  then
18:         $\mathbf{T} \leftarrow q_{new}$ 
19:         $\mathbf{Q}_{near}, q_{min} \leftarrow \text{BestParent}(\mathbf{T}, q_{new}, R^*)$ 
20:         $\mathbf{T} \leftarrow \text{Rewiring}(\mathbf{T}, \mathbf{Q}_{near}, q_{new}, q_{min}, R^*)$ 
21:      end if
22:    end if
23:  end while
24:   $N_{path}, t_{cal} \leftarrow \text{SelectPath}(\mathbf{T}, q_f)$ 
25:  return  $N_{path}, t_{cal}$ 
26: end procedure

```

3.1. Dubins Path Generation

Path generation is realized based on the Dubins path. A route is created by considering constant altitude, cruising speed v , and the UAV with maximum rotational curvature c_{\max} . Given the initial point P_s and final point P_f , the shortest path includes a combination of three path segments: a straight line segment (S) and an arc segment with a minimum radius (R or L). The four cases, LSL, LSR, RSR, and RSL, of the Dubins path are composed of two curved segments and a straight segment [20]. The shortest path is selected by comparing the four generated paths. The geometry of the Dubins path is depicted in Figure 6.

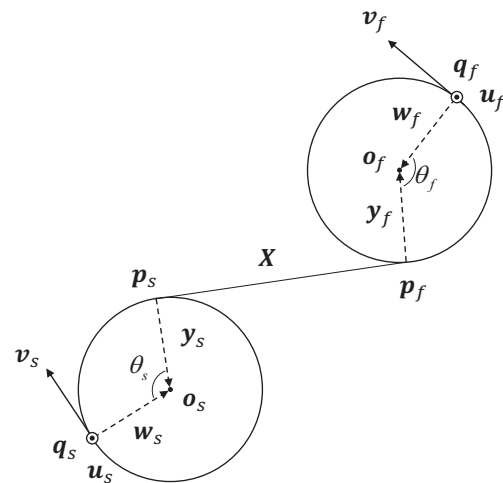


Figure 6. Geometry of the CSC path.

The positions and directions of the initial and final points in a 3D space are defined as

$$q_s = [x_{sx} \ x_{sy} \ x_{sz}]^T \tag{1}$$

$$q_f = [x_{fx} \ x_{fy} \ x_{fz}]^T \tag{2}$$

$$v_s = [v_{sx} \ v_{sy} \ v_{sz}]^T \tag{3}$$

$$v_f = [v_{fx} \ v_{fy} \ v_{fz}]^T \tag{4}$$

In the earth-fixed cartesian coordinate frame, the path is expressed as

$$\frac{dx(s)}{ds} = \cos \gamma(s) \cos \psi(s) \tag{5}$$

$$\frac{dy(s)}{ds} = \cos \gamma(s) \sin \psi(s) \tag{6}$$

$$\frac{dz(s)}{ds} = \sin \gamma(s) \tag{7}$$

$$\frac{d\psi(s)}{ds} = \eta \tag{8}$$

$$\frac{d\gamma(s)}{ds} = \mu \tag{9}$$

where s represents the curvilinear abscissa on the path, ψ is the heading angle, and γ is the flight path angle. As the vehicle is constrained by its minimum turn radius r , the path should have a maximum curvature limit c_{max} . The curvature $c(s)$ is given by

$$c(s) = \sqrt{\eta^2(s) \cos^2 \gamma(s) + \mu^2(s)} \tag{10}$$

and the objective is to minimize the path length while satisfying the following constraint:

$$-c_{max} \leq c(s) \leq c_{max} \tag{11}$$

Common to both planes of the initial and final curved paths of the minimum radius is a myriad of different planes, as shown in Figure 7.

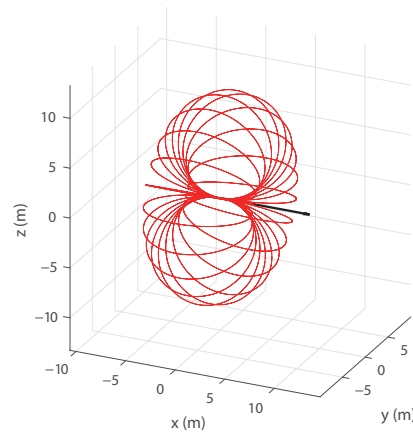


Figure 7. Torus geometry.

The straight-line path segment between these two curves is the intersecting line between the initial and final maneuver planes. Suppose $\mathbf{X}(X_x, X_y, X_z)$ is the vector common to both planes as follows [17]:

$$\mathbf{X} = \mathbf{p}_f - \mathbf{p}_s = (\mathbf{q}_f + r\mathbf{w}_f - r\mathbf{y}_f) - (\mathbf{q}_s + r\mathbf{w}_s - r\mathbf{y}_s) \tag{12}$$

Adding the first and second rotation angles θ_1 and θ_2 yields the following equation:

$$\mathbf{X} = (\mathbf{q}_f - \mathbf{q}_s) \mp r(\mathbf{x} + \mathbf{v}_s) \tan \frac{\theta_s}{2} \mp r(\mathbf{x} + \mathbf{v}_f) \tan \frac{\theta_f}{2} \tag{13}$$

where $\cos \theta_s = \mathbf{v}_s \mathbf{v}$ and $\cos \theta_f = \mathbf{v}_f \mathbf{x}$, and \mathbf{x} are unit vectors of \mathbf{X} . Through simplification, one obtains the following equation:

$$(x_{sx} - x_{fx}) = X_x \pm rx_x \left[\tan \frac{\theta_s}{2} + \tan \frac{\theta_f}{2} \right] \pm r \left[v_{fx} \tan \frac{\theta_f}{2} + v_{sx} \tan \frac{\theta_s}{2} \right] \tag{14}$$

$$(x_{sy} - x_{fy}) = X_y \pm rx_y \left[\tan \frac{\theta_s}{2} + \tan \frac{\theta_f}{2} \right] \pm r \left[v_{fy} \tan \frac{\theta_f}{2} + v_{sy} \tan \frac{\theta_s}{2} \right] \tag{15}$$

$$(x_{sz} - x_{fz}) = X_z \pm rx_z \left[\tan \frac{\theta_s}{2} + \tan \frac{\theta_f}{2} \right] \pm r \left[v_{fz} \tan \frac{\theta_f}{2} + v_{sz} \tan \frac{\theta_s}{2} \right] \tag{16}$$

In the case of a system of multivariable equations, the solution of the system of equations can be determined through the Gauss–Newton method, among other methods. In this manner, a common plane \mathbf{X} is obtained.

3.2. Optimization Improvement Method

The proposed algorithm has two methods to complement optimality. The first method considers the constraint condition of the minimum turning radius when the positions and directions of the initial point and the final point are given and generates a path using the Dubins path that generates the shortest path.

- As shown in Figure 4a, $\mathbf{q}_{nearest}^*$ is selected by searching for the node closest to \mathbf{q}_f and \mathbf{q}_{new}^* is created at a certain distance ϵ from $\mathbf{q}_{nearest}^*$ in the \mathbf{q}_f direction (lines 4 and 5).
- \mathbf{q}^* means that \mathbf{q}_{new}^* is the closest point to the Dubins path. \mathbf{q}^* is the point where the paths of \mathbf{q}^* new and the Dubins path are vertical (line 6).
- As shown in Figure 4b, one selects $\mathbf{q}'_{nearest}$ by searching for the node closest to \mathbf{q}^* and creates \mathbf{q}'_{new} at a position ϵ away from $\mathbf{q}'_{nearest}$ by a certain distance ϵ in the \mathbf{q}^* direction (lines 7 and 8).

The second method re-selects the parent node and re-connects the tree, similar to the RRT*.

- As shown in Figure 2a, q'_{new} or q_{new} is re-elected as the parent node among q_{near} with the lowest path cost (lines 11 and 18).
- If the path cost when connecting with q'_{new} or q_{new} is lower than the existing path cost of q_{near} within a certain radius, as shown in Figure 2b, the tree is re-constructed (lines 12 and 19).

3.3. Convergence Improvement Method

The proposed algorithm complemented the convergence by improving the biased RRT. The biased RRT is stochastically oriented to q_f , which causes unnecessary computation. Therefore, one always sets q_f to the sample node to improve convergence. However, if there are obstacles, there is a risk of falling into the local minima. To reduce this risk, sample nodes are selectively used.

- When selecting $q_{nearest}^*$, one proceeds with the sample node using q_f and generates q_{new}^* in the q_f direction (lines 4 and 5).
- As shown in Figure 5, when q'_{new} collides with an obstacle, $q_{nearest}$ is selected using q_{rand} and q_{new} is created (lines 3 and 13–15).

4. Path Tracking and Control

4.1. Path Tracking

A nonlinear UAV path tracking algorithm similar to the line-of-sight guidance algorithm is adopted. Equations of motion that do not take into account the dynamic characteristics of the UAV are expressed as

$$\dot{x} = v \cos \psi \cos \gamma \tag{17}$$

$$\dot{y} = v \sin \psi \cos \gamma \tag{18}$$

$$\dot{z} = v \sin \gamma \tag{19}$$

where v is assumed to be constant, and ψ and γ represent the control inputs. The path to be tracked is represented by the following equations:

$$\frac{dx_r(s)}{ds} = \cos \psi_r(s) \cos \gamma_r(s) \tag{20}$$

$$\frac{dy_r(s)}{ds} = \sin \psi_r(s) \cos \gamma_r(s) \tag{21}$$

$$\frac{dz_r(s)}{ds} = \sin \gamma_r(s) \tag{22}$$

At each time instant, s^* represents the path point closest to UAV in abscissa. This point ensures that the vector in the direction of the path from the UAV and the vector tangent to the path are perpendicular. Thus,

$$(x_r(s^*) - x) \frac{dx_r(s)}{ds} \Big|_{s=s^*} + (y_r(s^*) - y) \frac{dy_r(s)}{ds} \Big|_{s=s^*} + (z_r(s^*) - z) \frac{dz_r(s)}{ds} \Big|_{s=s^*} = 0 \tag{23}$$

where

$$e_x = x_r(s^*) - x \tag{24}$$

$$e_y = y_r(s^*) - y \tag{25}$$

$$e_z = z_r(s^*) - z \tag{26}$$

$$e = \sqrt{e_x^2 + e_y^2 + e_z^2} \tag{27}$$

$$\psi_r^* = \psi_r(s^*) \tag{28}$$

$$\gamma_r^* = \gamma_r(s^*) \tag{29}$$

This expression can be rewritten as

$$e_x \cos \psi_r^* \cos \gamma_r^* + e_y \sin \psi_r^* \cos \gamma_r^* + e_z \sin \gamma_r^* = 0 \tag{30}$$

Consider the following straight line Ω tangent to the path at (x_r, y_r, z_r)

$$\Omega = \begin{bmatrix} x_l(s) \\ y_l(s) \\ z_l(s) \end{bmatrix} = \begin{bmatrix} x_r(s^*) + s \cos \psi_r^* \cos \gamma_r^* \\ y_r(s^*) + s \sin \psi_r^* \cos \gamma_r^* \\ z_r(s^*) + s \sin \gamma_r^* \end{bmatrix} \tag{31}$$

where $K_p > 0$ represents the prediction distance of the existing line-of-sight derivation algorithm [21], and $s = 1/K_p$. If $K_p \rightarrow 0$, it follows the direction of the tangent while maintaining a constant cross-track error. If $K_p \rightarrow \infty$, it moves toward the path point closest to the UAV. At each time instant, ψ and γ are chosen so as to satisfy the following:

$$\cos \psi_d = \frac{\tilde{e}_x}{\sqrt{\tilde{e}_x^2 + \tilde{e}_y^2}} \tag{32}$$

$$\sin \psi_d = \frac{\tilde{e}_y}{\sqrt{\tilde{e}_x^2 + \tilde{e}_y^2}} \tag{33}$$

$$\cos \gamma_d = \frac{\sqrt{\tilde{e}_x^2 + \tilde{e}_y^2}}{\tilde{e}} \tag{34}$$

$$\sin \gamma_d = \frac{\tilde{e}_z}{\tilde{e}} \tag{35}$$

where

$$\tilde{e}_x = x_l(s)|_{s=1/K_p} - x_u = e_x + \frac{1}{K_p} \cos \psi_r^* \cos \gamma_r^* \tag{36}$$

$$\tilde{e}_y = y_l(s)|_{s=1/K_p} - y_u = e_y + \frac{1}{K_p} \sin \psi_r^* \cos \gamma_r^* \tag{37}$$

$$\tilde{e}_z = z_l(s)|_{s=1/K_p} - z_u = e_z + \frac{1}{K_p} \sin \gamma_r^* \tag{38}$$

$$\tilde{e} = \sqrt{\tilde{e}_x^2 + \tilde{e}_y^2 + \tilde{e}_z^2} = \sqrt{e^2 + \frac{1}{K_p^2}} \tag{39}$$

The bank angle command generation can be realized from the equation for the centrifugal force as follows:

$$F_L \sin \phi = ma \tag{40}$$

$$F_L \cos \phi = mg \tag{41}$$

where F_L is the total lift force, m is the mass, g is the gravitational acceleration, and ϕ is the bank angle. Accordingly,

$$\tan \phi = \frac{a}{g} \rightarrow \tan \phi_d = \frac{a_{cmd}}{g} \tag{42}$$

where ϕ_d is the bank angle command, and a_{cmd} is the lateral acceleration [22], which is defined as

$$a_{cmd} = \frac{2V^2}{L} \sin \eta \tag{43}$$

where L is the look-ahead distance, and η is the angle between the velocity vector V and line L .

4.2. Constraint Sliding Mode Controller

A sliding mode controller involving a robust control scheme that can ensure control performance and stability even in the presence of uncertainty is used [23,24]. The designed controller imposes a constraint on the angular velocity by adding a saturation function to the existing sliding mode controller.

The proposed controller is named the constraint sliding mode controller (CSMC), with the sliding surface defined as

$$s = \omega + A \operatorname{sat}_{\xi}(q_e) \tag{44}$$

where ω represents the angular velocities of the body frame, q_e is the error quaternion, and $A = \operatorname{diag}(a_1, a_2, a_3)$. Here, $a_i > 0$, and $\xi = c_{\max}/a_i$.

Since the sliding surface designed in Equation (44) is $s = 0$, the state variable converges to the target point in the same way as the operating characteristics of the existing sliding mode. Note that there is a point where the equilibrium point changes in the sliding phase. The conditions for the saturation function are defined as

$$\operatorname{sat}(q_i) = \begin{cases} \xi, & \text{if } \xi < q_i \\ q_i, & \text{if } -\xi \leq q_i \leq \xi \\ -\xi, & \text{if } -\xi > q_i \end{cases} \tag{45}$$

which represents a saturation function that allows a smaller value to be selected through a variable between the values of each component q_i of the posture error vector and the threshold variable ξ .

The reaching law for satisfying Equation (44) is defined as [25]

$$\dot{s} = -c \operatorname{sgn}(s) |s|^\beta \tag{46}$$

While deriving the control input, the three-axis control input for the UAV posture considering the angular velocity limitation can be obtained by differentiating the equation with respect to time and arranging the resulting equations and equations of motion of the UAV as follows:

$$u = I^{-1} \left[-f - C \operatorname{sgn}(s) |s|^\beta + \omega^\times J \omega - a J \frac{d}{dt} \left\{ \operatorname{sat}_{\xi}(q_e) \right\} \right] \tag{47}$$

To validate the stability of the designed controller, the Lyapunov candidate function is defined as

$$V = \frac{1}{2} s^T J s \tag{48}$$

and the equation is differentiated with respect to time to ensure stability as follows:

$$\begin{aligned} \dot{V} &= \mathbf{s}^T \mathbf{J} \dot{\mathbf{s}} = \mathbf{s}^T \mathbf{J} \left(\dot{\boldsymbol{\omega}} + a \frac{d}{dt}(\text{sat}(q_e)) \right) \\ &= \mathbf{s}^T \left(-\boldsymbol{\omega} \times \mathbf{J} \boldsymbol{\omega} + \boldsymbol{\tau} + a \mathbf{J} \frac{d}{dt}(\text{sat}(q_e)) \right) \\ &= \mathbf{s}^T \left(-c \mathbf{J} \text{sgn}(\mathbf{s}) |\mathbf{s}|^\beta \right) \\ &< 0 \end{aligned}$$

This expression is negative definite, and thus, it can be concluded that the control system is Lyapunov stable.

5. Simulation Study

The simulation is compared with the RRT, RRT*, and biased RRT to verify the performance of the proposed algorithm. Each algorithm ends the simulation when it creates a tree within a certain radius of q_f or ends when the number of trees is 1000. Note that the proposed algorithm applies the Dubins path to generate a path that takes into account the direction of flight and the minimum turning radius. To compare algorithms under the same conditions as the proposed algorithm, torus structures are set as obstacles at the initial and final points, as shown in Figure 7, and paths are generated considering flight direction and minimum radius of rotation. For the reasonable analysis, it was simulated 100 times. For each algorithm, ϵ is 30 m, and the simulation termination condition $R_{threshold}$ is 50 m. The R^* of the RRT* and the proposed algorithm is $\epsilon \times 8$ m, and k of the biased RRT is set to 10%. Since short and fast path generation is important, the simulation analysis is made with respect to the number of tree generations, path length, and calculation time. In this work, the concept of average precision is used to quantitatively compare the performance of two different algorithms for performance analysis [13].

In the 3D space, one proceeded with the simulation in three cases, as shown in Figure 8.

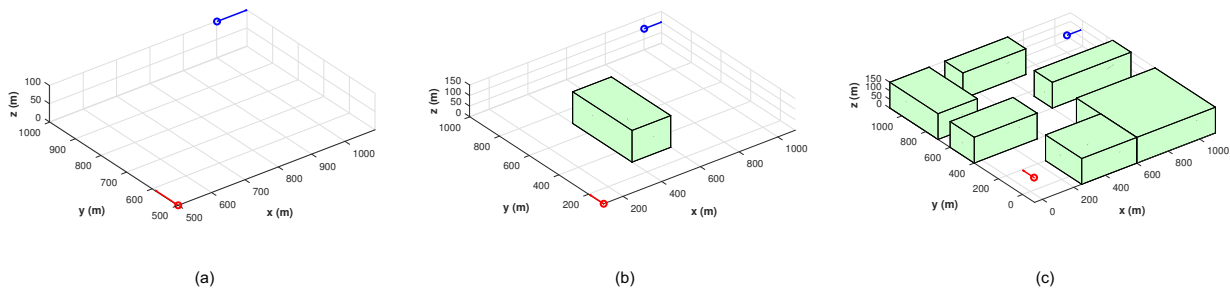


Figure 8. Simulation environment: (a) No obstacles. (b) Centered obstacle. (c) Scattered obstacles.

The states of the position and direction of each obstacle case are shown in Table 1. The performance of the proposed algorithm is verified using the fixed-wing UAV with a minimum turning radius r of 100 m as a sliding mode controller considering the guidance algorithm and angular velocity limit [26].

Table 1. Way-points for simulation.

No Obs				Center Obs/Scattered Obs			
q_s	q_f	v_s	v_f	q_s	q_f	v_s	v_f
$\begin{bmatrix} 500 \\ 500 \\ 0 \end{bmatrix}$ m	$\begin{bmatrix} 1000 \\ 1000 \\ 100 \end{bmatrix}$ m	$\begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}$	$\begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}$	$\begin{bmatrix} 100 \\ 100 \\ 0 \end{bmatrix}$ m	$\begin{bmatrix} 1000 \\ 1000 \\ 100 \end{bmatrix}$ m	$\begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}$	$\begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}$

5.1. Simulation Analysis (100 Times)

Figures 9–11 show each algorithm’s result performed 100 times for the three cases. Here, (a,e,i,m) is the path generation result, (b,f,j,n) is the total number of nodes, (c,g,k,o) is the path length, and (d,h,l,p) is the calculation time.

The existing algorithms encountered a local minima with about 51% no obstacles, about 53% for center obstacles, and about 49% for scattered obstacles. Further, the minimum radius of rotation has not been taken into account. On the other hand, the proposed algorithm considered the flight direction and the minimum rotation radius yielding a local minima-free path. However, there is a risk of falling into the local minima in the case of a space where it is unable to turn correctly because the path suggested is generated by considering the minimum dynamic turning radius of UAVs.

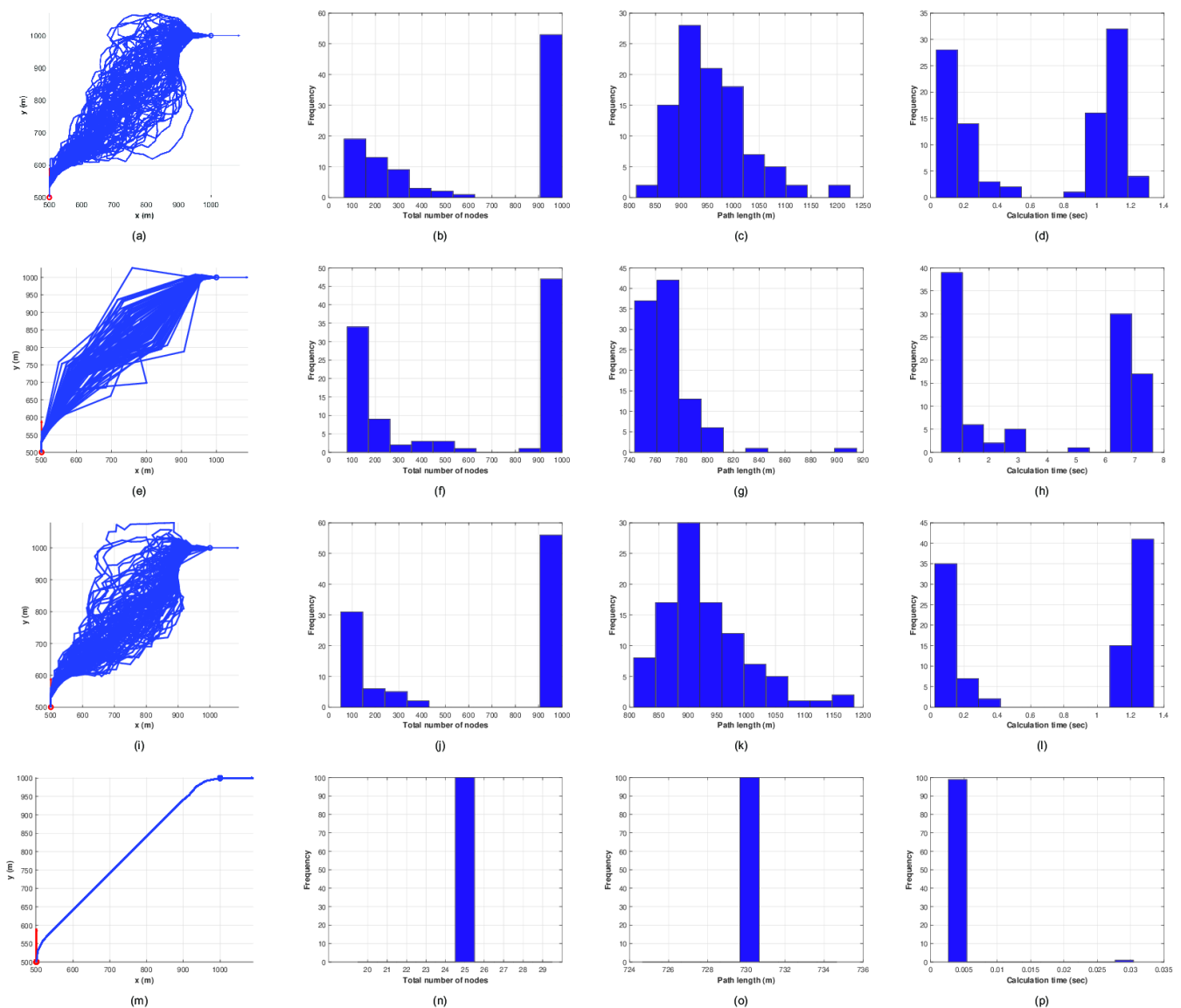


Figure 9. Results of no obstacles: (a–d) RRT. (e–h) RRT*. (i–l) Biased RRT. (m–p) Dubins RRT*.

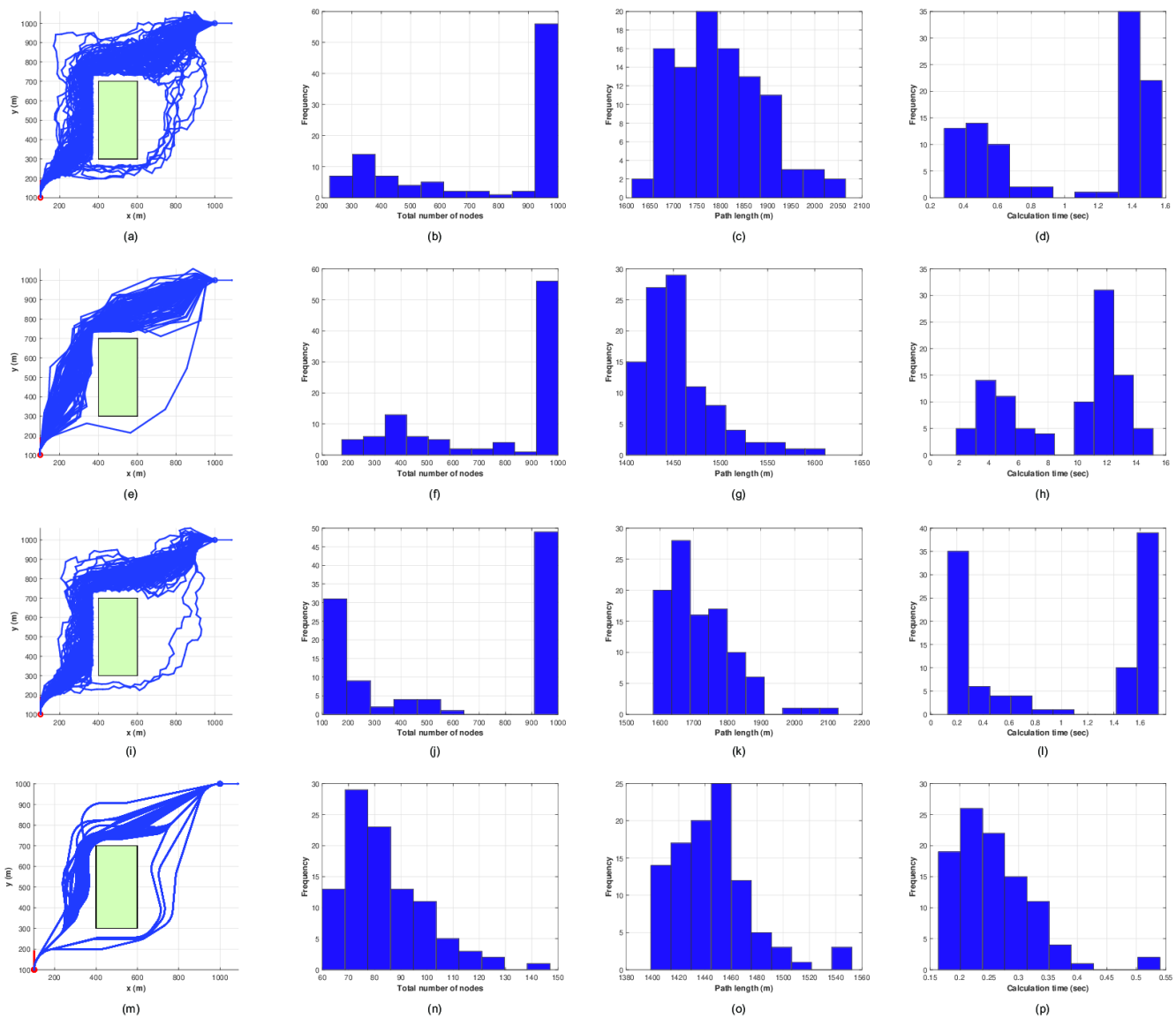


Figure 10. Results of center obstacle: (a–d) RRT. (e–h) RRT*. (i–l) Biased RRT. (m–p) Dubins RRT*.

Figure 12 represents the summary of the results. In all the cases, the proposed algorithms generated the overwhelmingly fewest nodes and generated paths. Further, the proposed algorithm for all obstacle cases showed the least computational burden. The proposed algorithm reduced the calculation time with respect to other algorithms in the no obstacles case by about 99.6%. In the case of the center obstacle, the calculation time of the proposed algorithm with respect to the RRT, RRT*, and biased RRT were reduced by 75.2%, 97.2%, and 72.6%, respectively. In the case of the scattered obstacles, the calculation time of the proposed algorithm with respect to the RRT, RRT*, and biased RRT were reduced by 27.6%, 91.8%, and 26.1%, respectively.

It is important to note that the proposed algorithm produced the shortest path in all obstacle cases. In the case of no obstacle, the path length using the proposed algorithm with respect to the RRT, RRT*, and biased RRT were reduced by 23.8%, 5%, and 21.7%, respectively. In the case of the center obstacle, the path length using the proposed algorithm with respect to the RRT, RRT*, and biased RRT were reduced by 19.6%, 0.5%, and 15.9%, respectively. In the case of the scattered obstacles, the path length using the proposed algorithm with respect to the RRT, RRT*, and biased RRT were reduced by 21.4%, 2%, and 18%, respectively.

It was confirmed that the proposed algorithm improved the performance effectively in both path length and calculation time compared to the three widely used algorithms, and the path was generated in consideration of the flight direction and minimum radius of rotation.

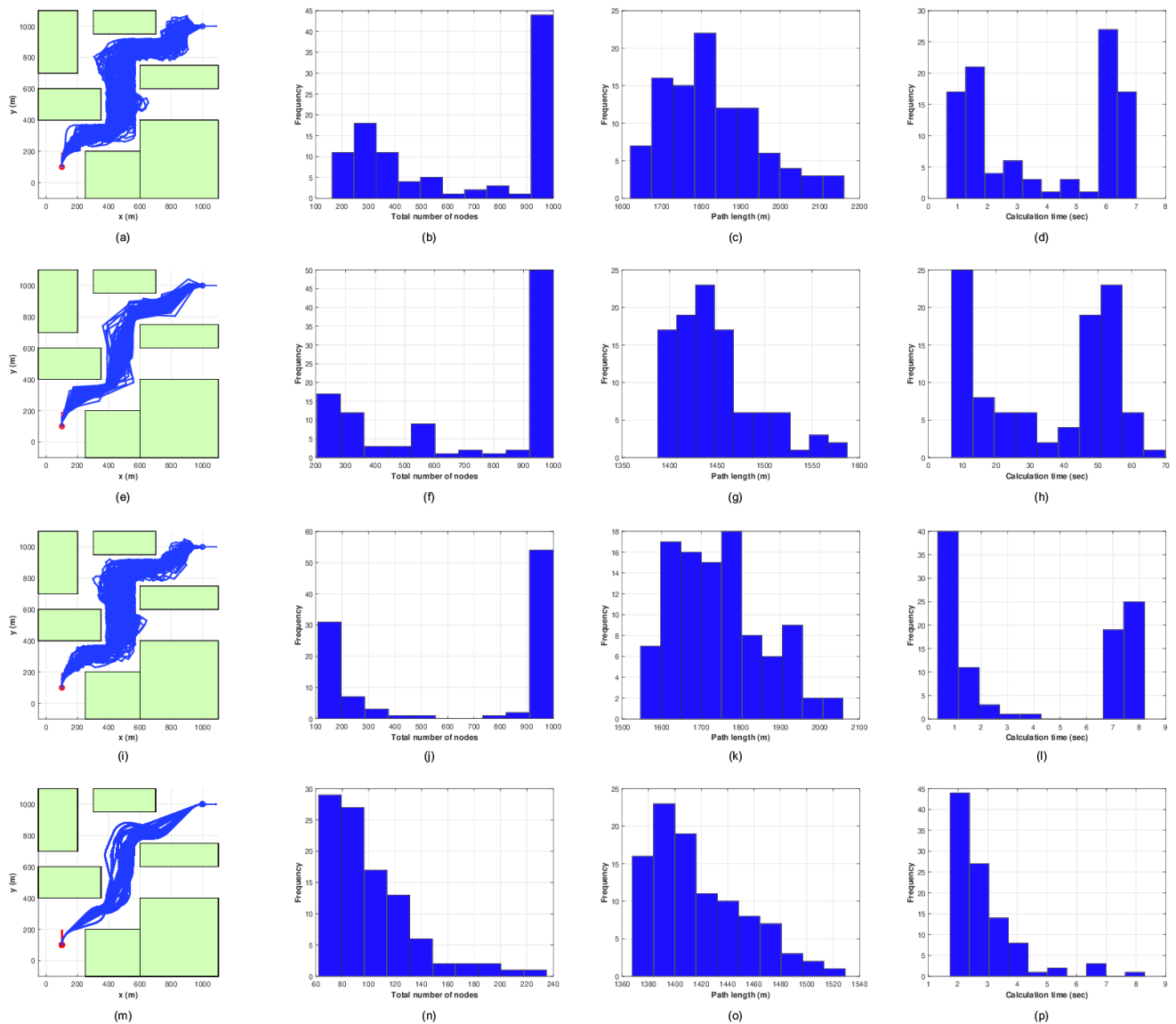


Figure 11. Results of scattered obstacles: (a–d) RRT. (e–h) RRT*. (i–l) Biased RRT. (m–p) Dubins RRT*.

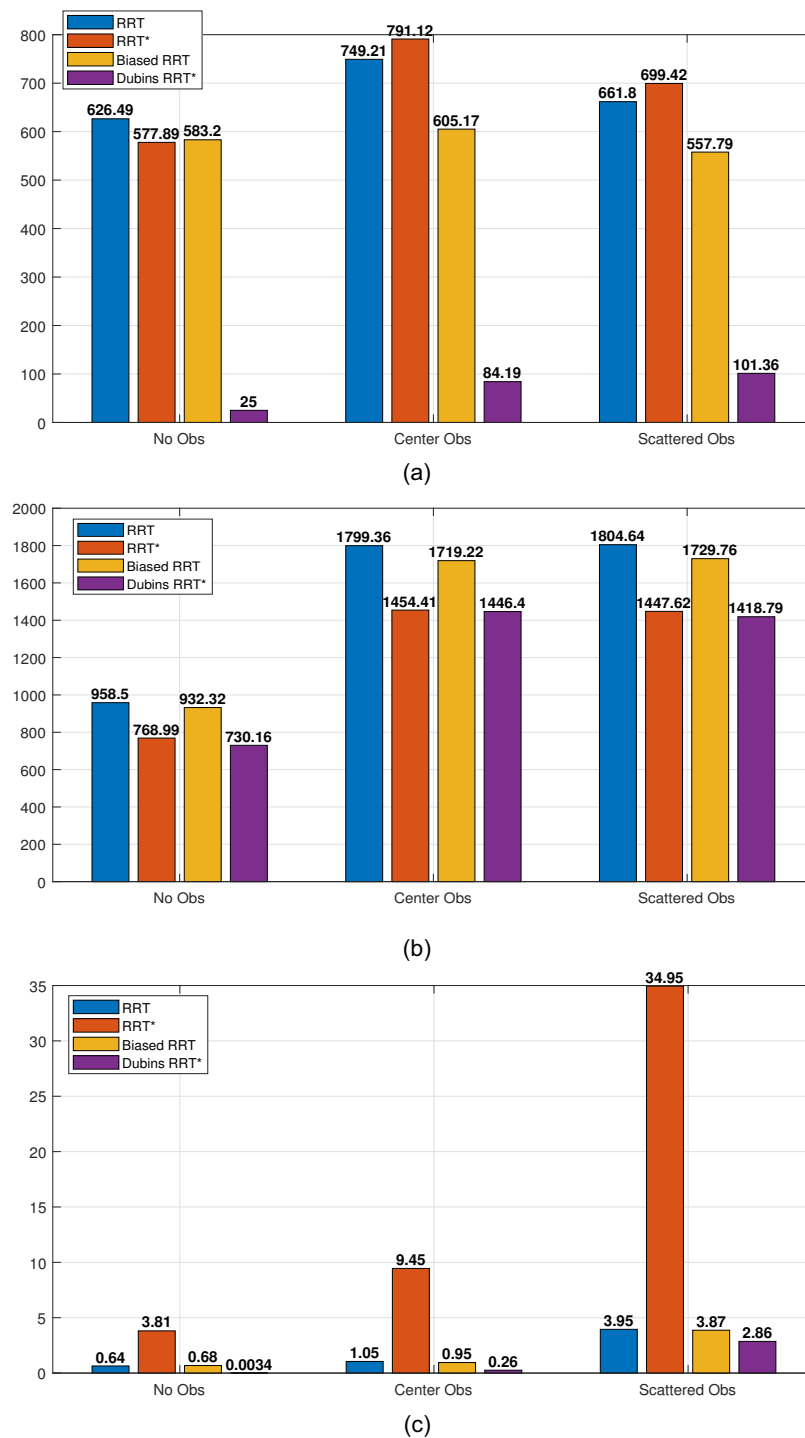


Figure 12. Results of 100 simulations: (a) Total number of nodes. (b) Path length. (c) Calculation time.

5.2. Simulation of UAV Path Tracking

Figure 13 shows the case of no obstacles. In the absence of obstacles, the path with the fastest computational time out of 100 simulations is generated in Figure 13a, and the generated path was tracked using a fixed-wing UAV. In the case of a position error, as shown in Figure 13b, the tracking performance is satisfactory, except in the beginning. An error occurs when the path is initiated in a circle; this error is likely a natural error caused by a sudden change in the attitude command due to the change in maneuvering. Nevertheless, the maximum position error is less than 5 m, and the straight line segment corresponds to an error of approximately 10^{-2} . Figure 13c shows the attitude command value generated

through the guidance algorithm and the actual attitude value of the UAV. As shown in Figure 13d, an error occurs during the circular maneuver, similar to the position error. This error is also likely a natural error due to the sudden change in the posture command due to the change in maneuvering, and an error of approximately 10^{-7} is observed in the straight line segment. Figure 13e shows the angular velocity of the UAV while satisfying the maximum angular velocity value range set by the CSMC, in accordance with the angular velocity limitation.

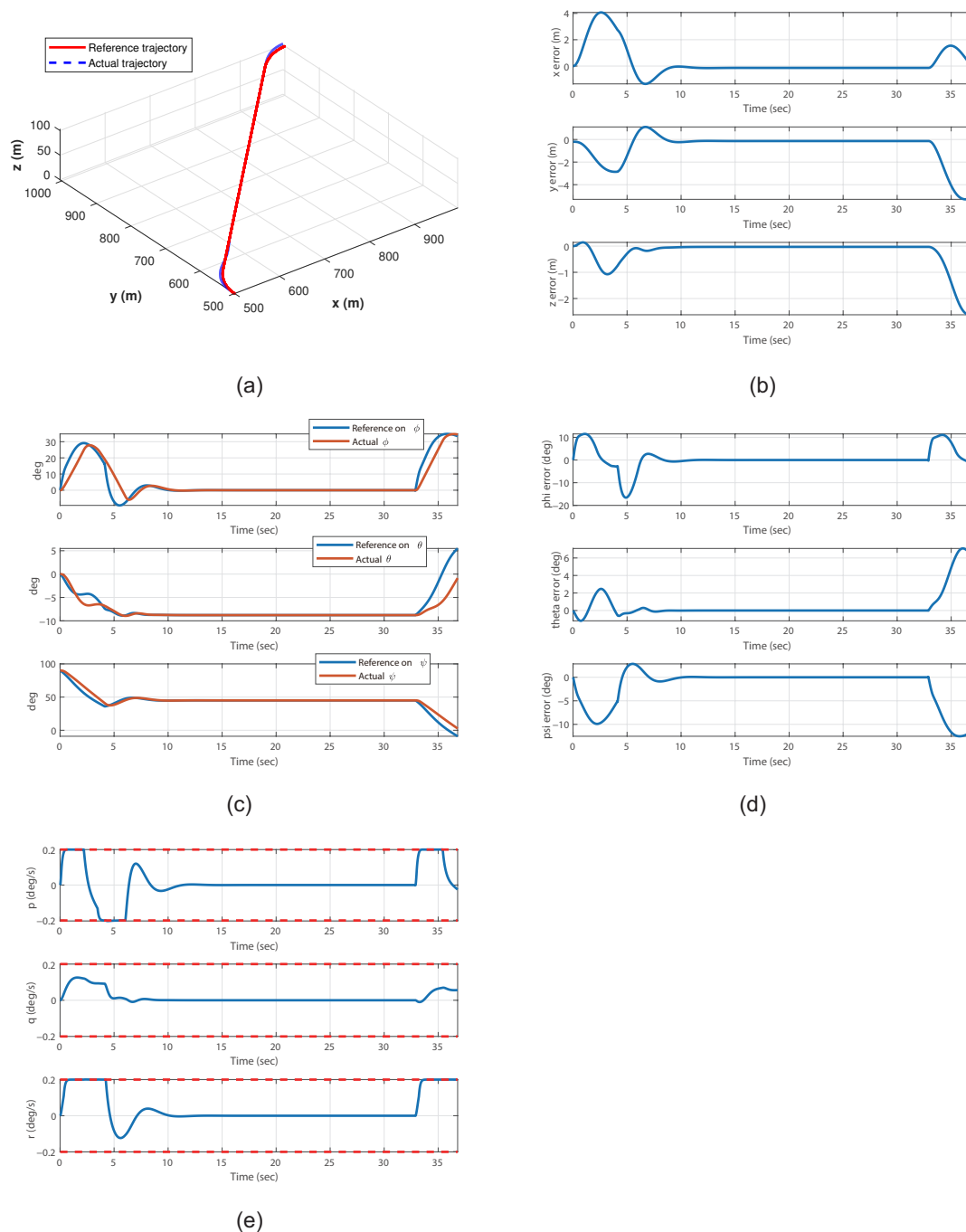


Figure 13. Path tracking results of no obstacles: (a) Path tracked in 3D. (b) Position error. (c) Attitude. (d) Attitude error. (e) Constrained angular velocity.

Figure 14 shows the case of the center obstacle. In the case of a center obstacle, the path with the fastest computational time out of 100 simulations was generated in Figure 14a, and the generated path was tracked using a fixed-wing UAV.

Figure 15 shows the case of scattered obstacles. In the case of scattered obstacles, the path with the fastest computational time out of 100 simulations was generated in Figure 15a, and the generated path was tracked using a fixed-wing UAV.

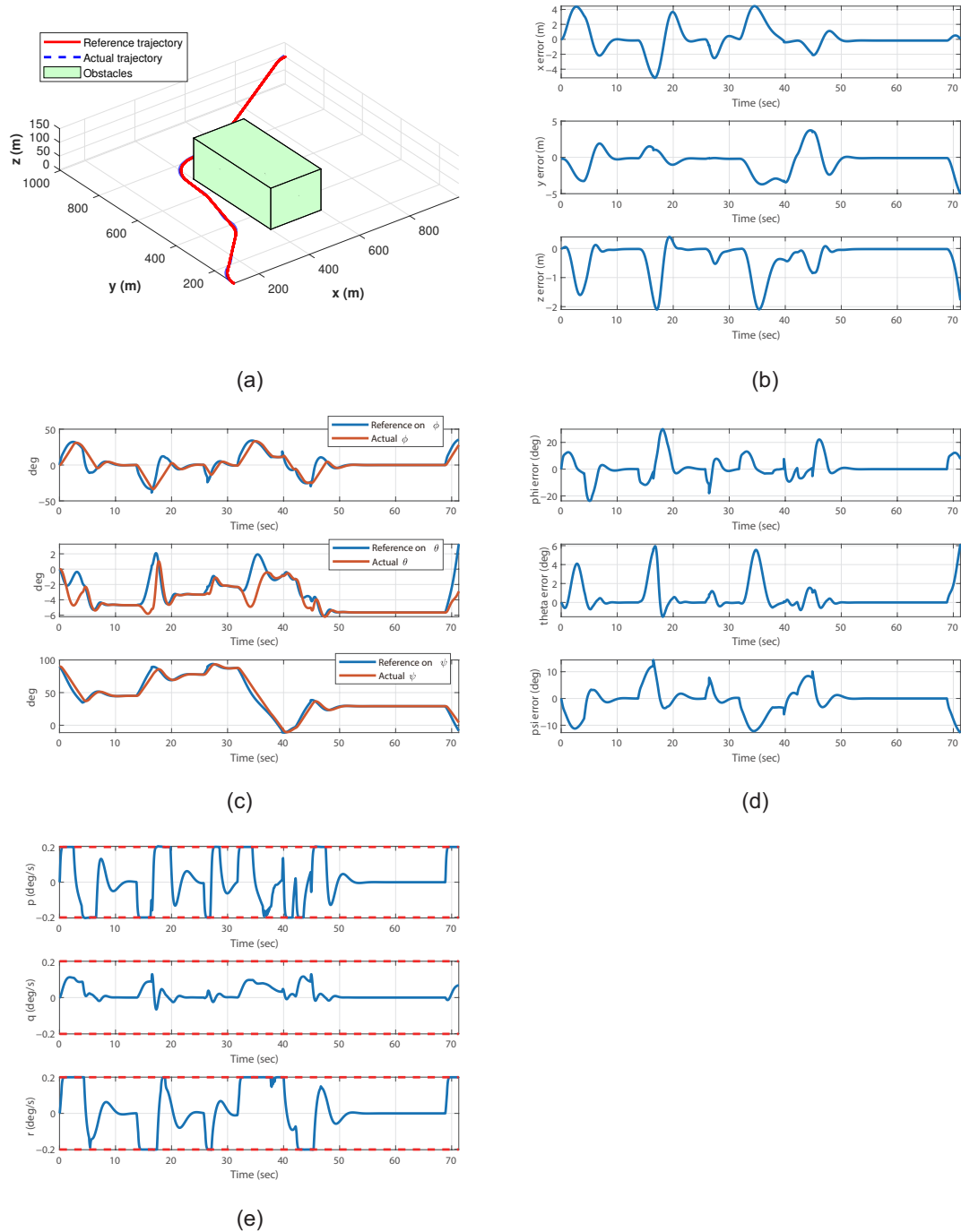


Figure 14. Path tracking results of center obstacle: (a) Path tracked in 3D. (b) Position error. (c) Attitude. (d) Attitude error. (e) Constrained angular velocity.

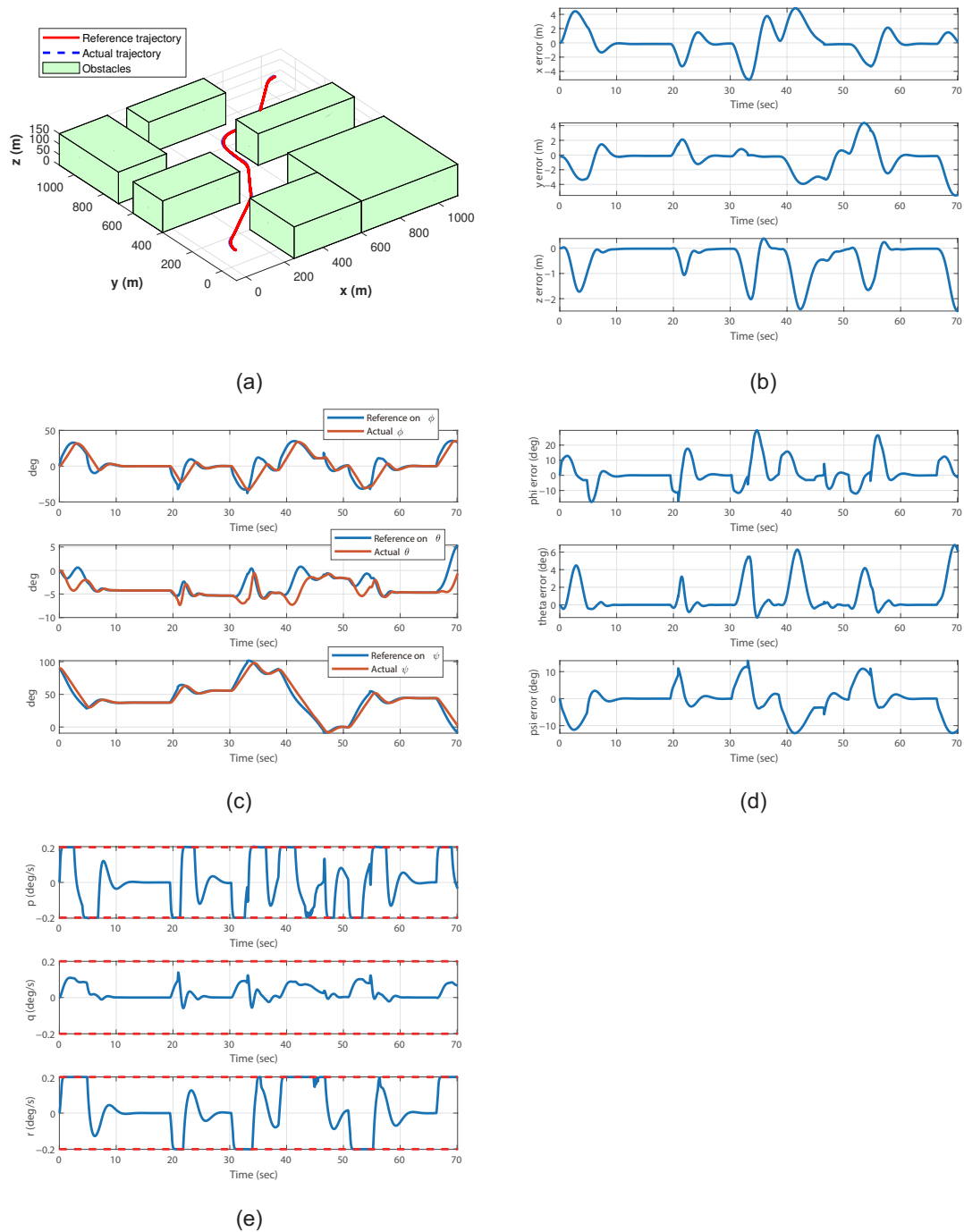


Figure 15. Path tracking results of scattered obstacle: (a) Path tracked in 3D. (b) Position error. (c) Attitude. (d) Attitude error. (e) Constrained angular velocity.

It is found that the result where obstacles exist is similar to the case where there is no obstacle. The maximum position error is within 5 m, and it showed an error of 10^{-2} when in a straight line segment. The maximum attitude error is within 23° , and it showed an error of 10^{-7} when in a straight line segment.

6. Conclusions

In this study, the authors solved the problem of generating a path to avoid obstacles to the target point without collision. This work proposes the Dubins path-oriented RRT* algorithm, which has improved optimality and convergence over the conventional RRT

algorithm and considers the flight direction and minimum turning radius according to the characteristics of the fixed-wing UAV. In order to improve optimality, the Dubins path was oriented, and the RRT* algorithm concept was added. In order to improve convergence, the target point was set as a sample node, and when there were obstacles, the randomly generated sample node was selectively set to solve the problem of convergence degradation. The performance of the proposed algorithm was verified by comparing 100 simulations of each algorithm under the same conditions. In the three obstacle cases, the proposed algorithm has improved path length and calculation time compared to the existing algorithm. Finally, using the guidance algorithm and the controller, one validated whether the path generated by the proposed algorithm followed the fixed-wing UAV. Consequently, the technology proposed in this paper is considered suitable for smooth missions with the safety and reliability of unmanned aerial vehicles and is suitable for actual applications. Future works will include experimental validation of the proposed algorithm.

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

Funding: This study was supported by research fund from Chosun University, 2022.

Data Availability Statement: No new data were created or analyzed in this study. Data sharing is not applicable to this article.

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

References

1. Wang, X.; Huang, Z.; Sui, G.; Lian, H. Analysis on the development trend of future UAV equipment technology. *Acad. J. Eng. Technol. Sci.* **2019**, *2*, 114–121. [[CrossRef](#)]
2. Shakhatareh, H.; Sawalmeh, A.H.; Al-Fuqaha, A.; Dou, Z.; Almaita, E.; Khalil, I.; Shamsiah, N. Unmanned aerial vehicles (UAVs): A survey on civil applications and key research challenges. *IEEE Access* **2019**, *7*, 48572–48634. [[CrossRef](#)]
3. He, S.; Chen, X.; Hung, M.; Chen, X.; Ji, Y. Steering angle measurement of UAV navigation based on improved image processing. *J. Inf. Hiding Multim. Signal Process.* **2019**, *10*, 384–391.
4. Lin, G.; Zhu, L.; Li, J.; Zou, X.; Tang, Y. Collision-free path planning for a guava-harvesting robot based on recurrent deep reinforcement learning. *Comput. Electron. Agric.* **2021**, *188*, 106350. [[CrossRef](#)]
5. Cao, X.; Yan, H.; Huang, Z.; Ai, S.; Xu, Y.; Fu, R.; Zou, X. A multi-objective particle swarm optimization for trajectory planning of fruit picking manipulator. *Agronomy* **2021**, *11*, 2286. [[CrossRef](#)]
6. Yang, L.; Qi, J.; Song, D.; Xiao, J.; Han, J.; Xia, Y. Survey of robot 3D path planning algorithms. *J. Control. Sci. Eng.* **2016**, *2016*, 7426913. [[CrossRef](#)]
7. Dijkstra, E. A note on two problems in connexion with graphs. *Numer. Math.* **1959**, *1*, 269–271. [[CrossRef](#)]
8. Wang, H.; Yu, Y.; Yuan, Q. Application of dijkstra algorithm in robot path-planning. In Proceedings of the 2011 Second International Conference on Mechanic Automation and Control Engineering IEEE, Hohhot, China, 15–17 July 2011; pp. 1067–1069.
9. Elbanhawi, M.; Milan, S. Sampling-based robot motion planning: A review. *IEEE Access* **2014**, *2*, 56–77. [[CrossRef](#)]
10. Karaman, S.; Frazzoli, E. Sampling-based algorithms for optimal motion planning. *Int. J. Robot. Res.* **2011**, *30*, 846–894. [[CrossRef](#)]
11. Anderson, S.J.; Peters, S.C.; Pilutti, T.E.; Iagnemma, K. An optimal-control-based framework for trajectory planning, threat assessment, and semi-autonomous control of passenger vehicles in hazard avoidance scenarios. *Int. J. Veh. Auton. Syst.* **2010**, *8*, 190–216. [[CrossRef](#)]
12. Paden, B.; Cap, M.; Yong, S.Z.; Yershov, D.; Frazzoli, E. A survey of motion planning and control techniques for self-driving urban vehicles. *IEEE Trans. Intell. Veh.* **2016**, *1*, 33–55. [[CrossRef](#)]
13. Karaman, S.; Walter, M.R.; Perez, A.; Frazzoli, E.; Teller, S. Anytime motion planning using the RRT. In Proceedings of the 2011 IEEE International Conference on Robotics and Automation, Shanghai, China, 9–13 May 2011; pp. 1478–1483.
14. Panice, G.; Luongo, S.; Gigante, G.; Pascarella, D.; Di Benedetto, C.; Vozella, A.; Pescapè, A. A SVM-based detection approach for GPS spoofing attacks to UAV. In Proceedings of the 2017 23rd International Conference on Automation and Computing (ICAC), Huddersfield, UK, 7–8 September 2017; pp. 1–11.
15. Noh, G.; Park, J.; Han, D.; Lee, D. Selective goal aiming rapidly exploring random tree path planning for UAVs. *Int. J. Aeronaut. Space Sci.* **2021**, *22*, 1397–1412.
16. Yang, K. Anytime synchronized-biased-greedy rapidly-exploring random tree path planning in two dimensional complex environments. *Int. J. Control. Autom. Syst.* **2011**, *9*, 750–758. [[CrossRef](#)]

17. Hota, S.; Ghose, D. Optimal geometrical path in 3D with curvature constraint. In Proceedings of the 2010 International Conference on Intelligent Robots and System, Taipei, Taiwan, 18–22 October 2010; Volume 79, pp. 113–118.
18. LaValle, S.M. Rapidly-Exploring Random Trees: A New Tool for Path Planning. 1998. Available online: <http://msl.cs.illinois.edu/~lavalle/papers/Lav98c.pdf> (accessed on 20 July 2022)
19. Noreen, I.; Khan, A.; Habib, Z. Optimal path planning using RRT* based approaches: A survey and future directions. *Int. J. Adv. Comput. Sci. Appl.* **2016**, *7*, 97–107. [[CrossRef](#)]
20. Manyam, S.G.; Casbeer, D.; Von Moll, A.L.; Fuchs, Z. Shortest Dubins Path to a Circle. In Proceedings of the AIAA Scitech 2019, San Diego, CA, USA, 7–11 January 2019.
21. Sujit, P.B.; Saripalli, S.; Sousa, J.B. Unmanned aerial vehicle path following: A survey and analysis of algorithms for fixed-wing unmanned aerial vehicles. *IEEE Control. Syst. Mag.* **2014**, *34*, 42–59.
22. Zhai, R.; Zhou, Z.; Zhang, W.; Sang, S.; Li, P. Control and navigation system for a fixed-wing unmanned aerial vehicle. *AIP Adv.* **2014**, *4*, 031306. [[CrossRef](#)]
23. Shtessel, Y.; Edwards, C.; Fridman, L.; Levant, A. *Sliding Mode Control and Observation*; Springer: New York, NY, USA, 2014; Volume 10.
24. Jang, S.H.; Yang, Y.Y.; Leeghim, H. Performance analysis for quadrotor attitude control by super twisting algorithm. *J. Korean Soc. Aeronaut. Space Sci.* **2020**, *48*, 373–381.
25. Xiong, J.J.; Zheng, E.H. Position and attitude tracking control for a quadrotor UAV. *ISA Trans.* **2014**, *53*, 725–731. [[CrossRef](#)]
26. Yang, Y.Y.; Leeghim, H. Dubins path generation and tracking in 3D for UAVs. In Proceedings of the Asia-Pacific International Symposium on Aerospace Technology, Jeju, Korea, 15–17 November 2021.