applied sciences

MDPI

Article

A Method of Enhancing Rapidly-Exploring Random Tree Robot Path Planning Using Midpoint Interpolation

Jin-Gu Kang [1], Yong-Sik Choi [2] and Jin-Woo Jung [1,*]

[1] Department of Computer Science and Engineering, Dongguk University, Seoul 04620, Korea; kanggu12@dongguk.edu
[2] Department of Artificial Intelligence, Dongguk University, Seoul 04620, Korea; sik2230@dongguk.edu
* Correspondence: jwjung@dongguk.edu; Tel.: +82-2-2260-3812

Abstract: It is difficult to guarantee optimality using the sampling-based rapidly-exploring random tree (RRT) method. To solve the problem, this paper proposes the post triangular processing of the midpoint interpolation method to minimize the planning time and shorten the path length of the sampling-based algorithm. The proposed method makes a path that is closer to the optimal path and somewhat solves the sharp path problem through the interpolation process. Experiments were conducted to verify the performance of the proposed method. Applying the method proposed in this paper to the RRT algorithm increases the efficiency of optimization by minimizing the planning time.

Keywords: robot path planning; RRT; midpoint interpolation; triangular rewiring; path smoothness

check for updates

Citation: Kang, J.-G.; Choi, Y.-S.; Jung, J.-W. A Method of Enhancing Rapidly-Exploring Random Tree Robot Path Planning Using Midpoint Interpolation. *Appl. Sci.* **2021**, *11*, 8483. https://doi.org/10.3390/app11188483

Academic Editor: António Paulo Moreira

Received: 8 July 2021
Accepted: 9 September 2021
Published: 13 September 2021

1. Introduction

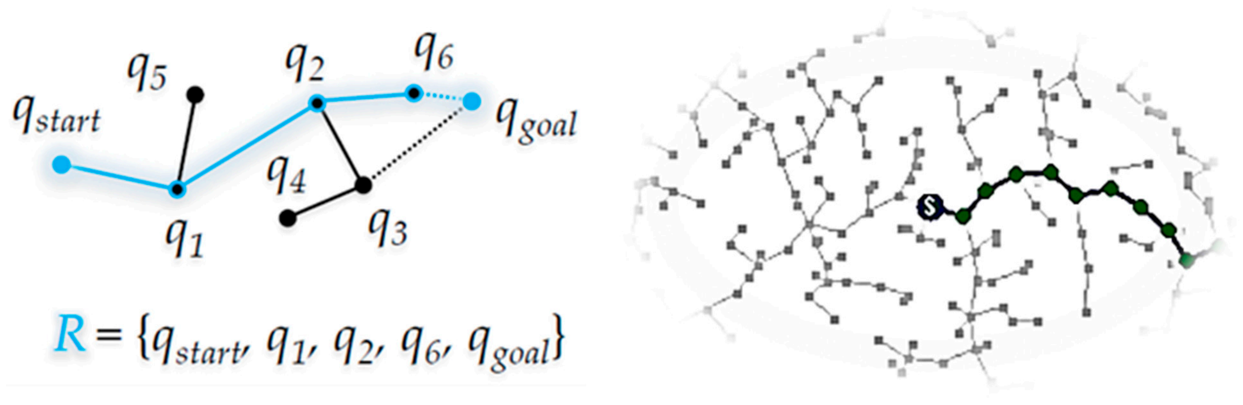
Recent path planning research for the robot has encompassed a wide range of topics [1,2]. Path planning is an important capability for autonomous mobile robots. A robot must be able to identify a path from its current position to its destination in order to move successfully. A mobile robot must be able to discover an optimal or sub-optimal collision-free path in the environment from the starting position to the destination [3].

Path planning is the formulation of a route for a mobile robot to proceed from a starting point to a destination point in Euclidean space as efficiently as possible while avoiding both static and dynamic obstacles and maintaining optimality, clearance, and completeness [4]. An optimal path is one with the ideal path length, a clear path is one without obstacles for the mobile robot to collide with, and a complete path is one in which the robot can move from the start point to the destination point without colliding with obstacles.

Furthermore, it is indeed possible for the robot to be able to optimize its path by determining the quickest and safest path to its destination point in order to save time and energy. However, an algorithm that generates the optimal path increases the computation, and an algorithm that quickly generates a path does not guarantee the optimal path [5].

It is difficult to ensure optimality with the sampling-based rapidly-exploring random tree (RRT) algorithm [6]. As shown in Figure 1a, the RRT algorithm is a path planning algorithm that involves repeatedly adding a randomly sampled position as a child node in a tree with the starting point as the root node until the destination point is reached. The tree extends out in the shape of a stochastic fractal, as shown in Figure 1b, and has an algorithm used to locate the destination point.

The RRT algorithm and other sampling-based algorithms [7,8] offer the benefit of planning a path in a shorter time with less computing than traditional path planning algorithms like the visibility graph- [9], cell decomposition- [10], and potential field-based algorithms [11]. On the other hand, it does not ensure optimality and has the drawback of having probabilistically assured completeness. The latter is also known as probabilistic completeness [12], which implies that completeness is assured when the number of random samples is infinite but not always when the number of random samples is limited. The goal of this research is to enhance the RRT algorithm, which ensures completeness and performs better than the related algorithms.



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

Figure 1. Overview of the rapidly-exploring random tree (RRT) algorithm: (**a**) planned path R from starting point q_{start} through waypoints q_1, q_2, q_6 to destination q_{goal} (q_i is a point on the path); (**b**) process of finding destination point by stochastic fractal shape from the root node (S) as a starting point.

To solve these problems, the main idea of the proposed post triangular processing of midpoint interpolation method is effective in path planning algorithms that do not guarantee optimality, such as the RRT algorithm that has a locally piecewise linear shape and can be used as a post-processing method after a path has been planned using one of these algorithms. It may also be used for different route planning methods since it is a post-processing technique that has no impact on calculation time.

The sampling-based path planning algorithm's primary strength is the fast planning speed based on the small amount of computation compared to the traditional path planning algorithms [7].

Performance verification using simulation in various environments and mathematical modeling were used to validate the performance of the proposed method in this paper. The case in which the proposed algorithm is applied to the sampling-based path planning method and the case in which it is not applied are compared through simulation. Here, the planning time and path length of the first complete path to reach a destination point from a starting point are evaluated.

This paper is organized as follows: Section 2 reviews some related works. Section 3 introduces the proposed post triangular processing of the midpoint interpolation method. Various experimental environments are constructed for path planning in Section 4 to examine the effectiveness and improvements of the proposed method. Finally, the conclusions are presented in Section 5.

2. Related Works

In this section, we introduce the previous works about the RRT algorithm in Section 2.1 and the Triangular Rewiring Method for the RRT Algorithm in Section 2.2, respectively.

2.1. RRT

This section shows the pseudocode of the RRT algorithm used in the experiment of this paper that was designed based on [6] in which the RRT algorithm was proposed. In 1998, LaValle proposed the RRT algorithm, which is a representative sampling-based path planning algorithm [6]. It is designed to have many degrees of freedom and is useful for planning a path under non-holonomic constraints.

As shown in Figure 2, when a random sample is generated in the configuration space, the node nearest to the position of the random sample is identified among the nodes constituting the tree with the starting point as the root node. A new node is generated at the random sample position and inserted into the tree if the random sample position is nearer than the step length. The process of tree extension is repeated until the destination point is reached. The RRT algorithm implemented for the proposed method and comparison experiment is Algorithm 1.

Algorithm 1 Pseudocode of RRT Algorithm

```
Input:
$q_{start} \leftarrow$ start point
$q_{goal} \leftarrow$ goal point
$\lambda \leftarrow$ step length
$C \leftarrow$ position set of all (measured) boundary points in all (known) obstacles
$N \leftarrow$ number of random samples
Output:
$R \leftarrow$ result of path $R$
Initialize:
$T \leftarrow$ Null tree<node, edge>
Procedure RRT
Begin
1     $T \leftarrow$ Insert root node<$q_{start}$> to $T$
2     While $n \leftarrow 0$ To $N$ Do
3             Generate $n$-th random sample
4         $q_{rand} \leftarrow$ position of $n$-th random sample
5         $q_{near} \leftarrow$ position of the nearest node in $T$ from $q_{rand}$
6         If not isInside($q_{near}$, $q_{rand}$, $\lambda$) Then
7             $q_{new} \leftarrow$ intersection point between line segment connecting $q_{rand}$ and $q_{near}$,
      and circle with radius $\lambda$ centered at $q_{near}$
8                 Else $q_{new} \leftarrow q_{rand}$
9             If not isTrapped($q_{new}$, $q_{near}$, C) Then
10            $T \leftarrow$ Insert node<$q_{new}$> and edge<$q_{new}$, $q_{near}$> to $T$
11        If isInside($q_{new}$, $q_{goal}$, $\lambda$) Then
12            $T \leftarrow$ Insert node<$q_{goal}$> and edge<$q_{new}$, $q_{goal}$> to $T$
13        $P \leftarrow$ path from last inserted node{$q_{goal}$} to root node{$q_{start}$} in $T$
14    If [length of $R$] > [length of $P$] Then    $R \leftarrow P$
15        $T \leftarrow$ Delete node<$q_{goal}$> and edge<$q_{new}$, $q_{goal}$> from $T$
End
```

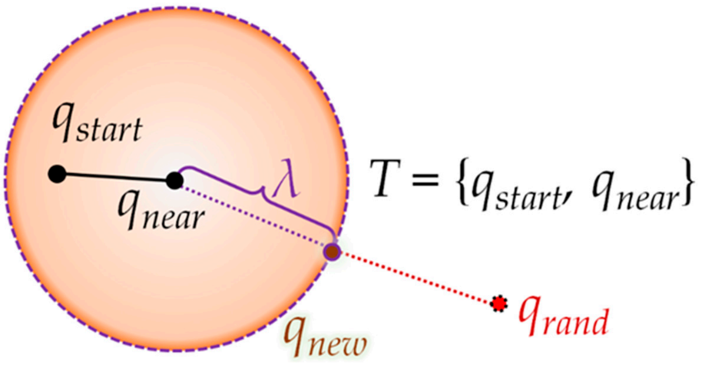


Figure 2. Process of the RRT algorithm: When creating the new node q_{new} at a position separated by step length λ in direction of random sample position (q_{rand}) based on q_{near} node (position) closest to random sample position (q_{rand}) in tree T with starting point q_{start} as the root node.

In order to overcome the limitations on optimality and convergence time [6], RRT-Connect can find a connected path more quickly by setting the start point and the destination point as the root of a separate tree, and further expanding the two trees alternately [7]. Rapidly-exploring Random Tree Star (RRT*) [13] was developed to overcome the limitation that the path generated from RRT does not guarantee convergence to the optimal path.

Informed-RRT* that can find a connected path quickly by enhancing the sampling probability inside the elliptical region with the start point and the destination point as the respective focal points [14]. The RRT*-Connect algorithm combines the advantages of RRT-Connect and RRT* [15]. RRT*-Smart [16], Quick-RRT*[17], and the proposed algorithm in [8] can show closer optimality by finding and connecting linearly connectable ancestor nodes to random samples through triangular inequality in the process of adding random samples.

2.2. Triangular Rewiring Method for the RRT Algorithm

This section shows the principle and pseudocode of the Triangular Rewiring Method for the RRT algorithm.

The triangular rewiring method is used to rewire the component based on the triangular inequality concept [8]. The triangular inequality-based RRT algorithm is a rewiring of the RRT method that is based on the concept of triangular inequality between nodes in path planning; thus, it is closer to the optimum than the RRT.

The triangular rewiring method not only can find a better initial solution but also can converge to a better solution rapidly.

The pseudocode for the triangular rewiring method is shown in Algorithm 2. This iterative process continues until no $q_{ancestor}$ exists (when no parent node exists for the previous $q_{ancestor}$, i.e., when $q_{ancestor}$ is q_{start}) or an obstacle exists between q_{child} and $q_{ancestor}$. The method for triangular rewiring is as follows: the node with the position $\mathrm{q}_{\mathrm{parent}}$ and the edge connecting the q_{child} and $q_{ancestor}$ nodes are deleted. After the edge is deleted, the $\mathrm{q}_{\mathrm{ancestor}}$ node is updated with the q_{parent} node, and the parent node of the $q_{ancestor}$ is updated with the $q_{ancestor}$ node. The existing q_{parent} node is then deleted. Then, the Trapped method is used to check if it collides with an obstacle between the q_{child} node and the updated q_{parent} node. Then, in tree T, the last created q_{parent} is inserted as the parent node of q_{child}.

Algorithm 2 Pseudocode of Triangular Rewiring Method for RRT Algorithm

```
Input:
$q_{child} \leftarrow$ Position $\{q_{new}/q_{newA}/q_{newB}\}$
$q_{parent} \leftarrow$ Position $q_{near}$
$T \leftarrow$ Tree $\{T_{merged}/T_a/T_b\}$
$C \leftarrow$ Position Set $C$
Output:
$\{T_{merged}/T_a/T_b\} \leftarrow$ Result of $T$
Begin triangular Rewiring Procedure
1     $q_{ancestor} \leftarrow$ Position of Parent Node of $q_{parent}$ in $T$
2     If Not isTrapped($q_{ancestor}, q_{child}, C$) then
3         $T \leftarrow$ Delete Node<$q_{parent}$>, Edge<$q_{parent}, q_{child}$> and Edge<$q_{parent}, q_{ancestor}$> from $T$
4         $q_{parent} \leftarrow q_{ancestor}$
5         $q_{ancestor} \leftarrow$ Position of Parent Node of $q_{ancestor}$ in $T$
6         While Not $q_{ancestor}$ = Null do
7             If Not isTrapped($q_{ancestor}, q_{child}, C$) then
8                 $T \leftarrow$ Delete Node<$q_{parent}$> and Edge<$q_{parent}, q_{ancestor}$> from $T$
9                 $q_{parent} \leftarrow q_{ancestor}$
10                $q_{ancestor} \leftarrow$ Position of Parent Node of $q_{ancestor}$ in $T$
11            Else
12                Break
13        $T \leftarrow$ Insert Edge<$q_{parent}, q_{child}$> to $T$
14    Else
15        $T \leftarrow$ Insert Node<$q_{child}$> and Edge<$q_{child}, q_{parent}$> to $T$
End triangular Rewiring Procedure
```

3. Proposed Post Triangular Processing of the Midpoint Interpolation Method

The proposed post triangular processing of the midpoint interpolation method can be applied to path planning algorithms that do not guarantee optimality, such as the RRT algorithm, and rewiring and midpoint interpolation based on the triangular inequality principle.

The assumptions of the proposed method are as follows:

1. The destination point may change gradually over time, but there is only one start point and one destination point for each tree.
2. If the mobile robot cannot perform the omnidirectional motion, local planning or kinodynamic planning is performed separately on the path planning result.

The basic principle is that the node serving as a waypoint in the planned path checks whether an obstacle collides with its own grandparent node, and if it is free from obstacle collision, the grandparent node rewires to the parent node. If it is not free from obstacle collision, as shown in Figure 3, the locally piecewise linear path created between the node, its parent node, and its grandparent node is made a more optimal path through the interpolation process. In this process, a new node is interpolated into the path and deviates from the piecewise linear path, so it has the advantage of being able to somewhat solve the sharp path problem (the problem facing a mobile robot that has kinematic constraints because the slope is not smooth).

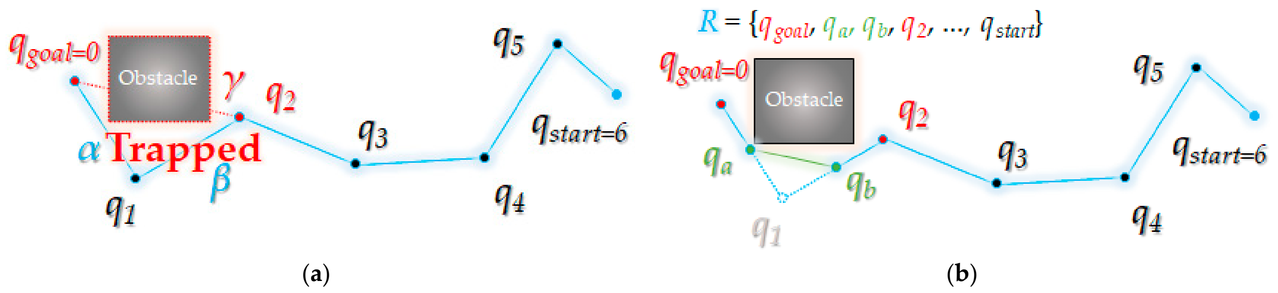


Figure 3. Summary of post triangular processing of the midpoint interpolation method: (**a**) line segment γ with node q_0 and its grandparent node q_2 in tree R is not free from obstacle collision; (**b**) Parent node q_1 of q_0 is deleted, and q_a between q_0 and q_1, and q_b between q_1 and q_2 are inserted as waypoints of a new path between q_0 and q_2.

This post triangular processing of the midpoint interpolation method was designed based on the polygon approximation algorithm [18,19], so, as shown in Figure 4, the path is calculated through a constant value called ε (the threshold of minimum clearance) ($\varepsilon > 0$). It determines how closely the obstacle is approximated or, in other words, how close to the optimal path it is. Here, in Figure 4, d follows Equation (1):

$$d_n(q_i) = \begin{cases} (d_{n-1}(q_i))/2, & n > 0 \\ \left(2\sqrt{s(s-\alpha)(s-\beta)(s-\gamma)}\right)/\gamma, & n = 0 \end{cases} \quad (s = (\alpha+\beta+\gamma)/2) \tag{1}$$

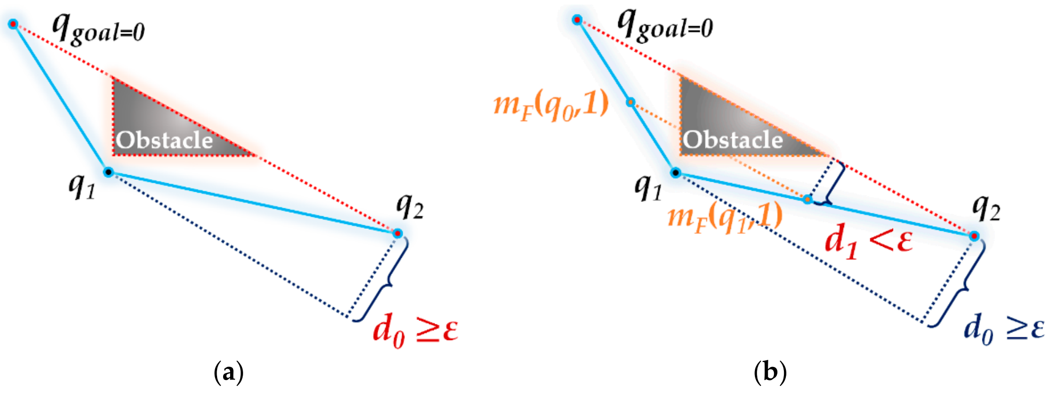


Figure 4. Interpolation function of post triangular processing of the midpoint interpolation method: (**a**) height d_0 of the triangle formed by waypoints q_0, q_1, and q_2 of the path is greater than ε (interpolation continuation); (**b**) height d_1 of the triangle formed by midpoints $m_F(q_0,1)$ and q_1 of q_0 and q_1 and midpoint $m_F(q_0,1)$ is the midpoint of q_0 and q_1, also $m_F(q_1,1)$ is the midpoint of q_1 and q_2.

Here, $\xi()$ is a function that receives the node as a variable and returns the parent node of that node. The n-squared ($n \geq 0$) of the $\xi()$ function can be expressed as $\xi^n(q_i) := \overbrace{(\xi \circ \xi \circ \ldots \circ \xi)}^{n}(q_i)$, and if n is 0, $\xi^0(q_i) := q_i$ holds.

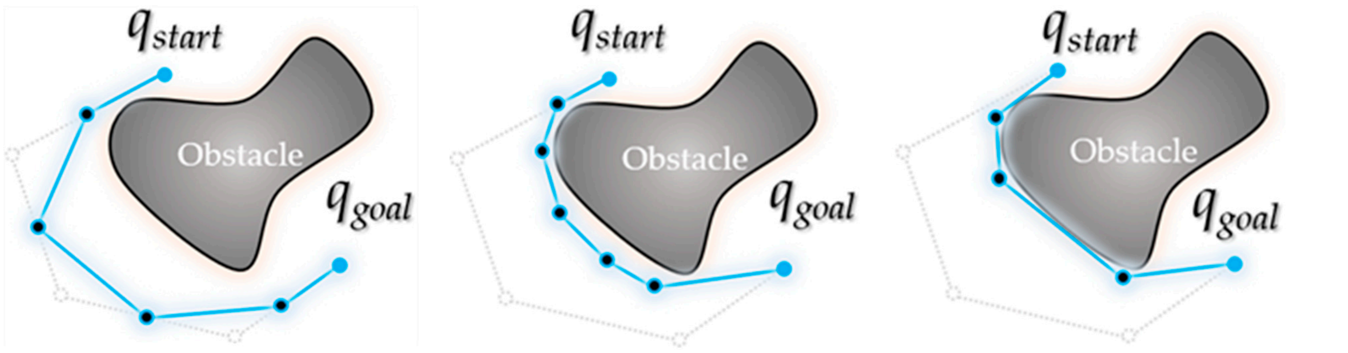
For the waypoint q_i, the value of d decreases by $1/2$ as interpolation proceeds (n). The initial value d_0 is the height of the triangle formed by the three line segments α, β, and γ ($\gamma < \alpha + \beta$), and the value of d_n becomes $(d_{n-1})/2$. Based on q_i, let α be the line segment from itself to the parent node, β be the line segment from the parent node to the grandparent node, and γ be the line segment from itself to the grandparent node. This d value serves as a measure to confirm the clearance of the obstacle compared to ε. This is because the smaller the d, the closer the path is to the obstacle.

Equation (1) converges to 0 when n becomes infinitely large in d_n, so epsilon receives only a value greater than 0 (positive number) from the user. ε is always greater than d_n when n is infinitely large. i.e., d is always smaller than epsilon at some point when n becomes infinitely large. This can be expressed as Equations (2) and (3) as follows:

$$\lim_{n\to\infty} d_n(q_i) = \lim_{n\to\infty} \frac{d_{n-1}(q_i)}{2} = \lim_{n\to\infty} \frac{d_0(q_i)}{2^n} = \lim_{n\to\infty} \frac{\mathbb{C}}{2^n} = 0, \tag{2}$$

$$\therefore \lim_{n\to\infty} d_n(q_i) = 0,\ \varepsilon > 0 \to \varepsilon > \lim_{n\to\infty} d_n(q_i). \tag{3}$$

However, since optimality and clearance are opposite attributes, the closer ε is to 0 as shown in Figure 5, the more similar the path or path length is to the visibility graph, but it is not a smooth path [20]. The farther away it is from 0 (within a significant value where the path is modified), the farther it is from the optimum, but a smooth (kinetic) path is made, so ε should be set to a suitable value according to the environment.



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

Figure 5. Variations according to ε values in post triangular processing of midpoint interpolation method (ε value of (**a**) > ε value of (**b**) > ε value of (**c**)): (**a**) When ε value is set relatively large; (**b**) When ε value is set relatively medium (smooth path); (**c**) When ε value is set relatively small (closer to the optimal path).

The following Algorithm 3 shows the pseudocode of the proposed post triangular processing of the midpoint interpolation method. It is mainly composed of the post triangular function (Algorithm 4) and interpolation function (Algorithm 5).

The input values of the post triangular processing of the midpoint interpolation method consist of the path R planned through a path planning algorithm, such as the RRT algorithm, the obstacle area information C, and the threshold value ε of the minimum clearance.

The f_{modify} is a variable that determines whether the input path R has been modified by this method, and if the path is modified even once, the entire process is repeated. If the path modification does not occur in the process of repeating, the algorithm is terminated. t refers to the index of the currently focused waypoint of R. That is, if t is 0, it is the starting point, which is the first point of R.

Algorithm 3 Pseudocode of Proposed Post Triangular Processing of Midpoint Interpolation Method

```
Input:
$R \leftarrow$ path from {RRT/ . . . }
$C \leftarrow$ position set of all (measured) boundary points in all (known) obstacles
$\varepsilon \leftarrow$ threshold value of minimum clearance
Output:
$R \leftarrow$ modified path $R$
Initialize:
$f_{modify} \leftarrow$ true
Procedure postTriProcOfMidInterpolation
Begin
1       While $f_{modify}$ Do
2           $f_{modify} \leftarrow$ false        // is the path modified
3           $t \leftarrow 0$        // index of the currently focused point
4           $q_{child} \leftarrow$ first point in $R$
5           $q_{parent} \leftarrow$ next point of $q_{child}$ in $R$
6           While not [$q_{parent}$ is the last point in $R$] Do
7               $q_{ancestor} \leftarrow$ next point of $q_{parent}$ in $R$
8               If not isTrapped($q_{child}$, $q_{ancestor}$, C) Then
10                  $R \leftarrow$ postTriangular($R$, $\varepsilon$, $t$, $f_{modify}$)
11              Else
12                  $R \leftarrow$ interpolation($R$, $C$, $\varepsilon$, $t$, $f_{modify}$)
13                  $q_{child} \leftarrow$ $t$-th point in R
14              $q_{parent} \leftarrow$ next point of $q_{child}$ in R
End
```

In R, when the first focusing point is q_{child}, the next point of that point q_{child} is q_{parent}, and the next point of that point q_{parent} is $q_{ancestor}$. It is determined whether the distance between q_{child} and $q_{ancestor}$ is free from obstacle collision (*isTrapped*() function). If it is free from collision, it is called *postTriangular*(); otherwise, it is called *interpolation*(). *postTriangular*() connects q_{child} and $q_{ancestor}$ as in the triangular rewiring method [8], and the q_{parent} between them is deleted from the path. *interpolation*() finds (interpolates) the points between q_{child} and q_{parent} and between q_{parent} and $q_{ancestor}$ that are free from obstacle collision when connected and rewires q_{child}, $q_{ancestor}$, and those two points are found. If R and t are updated due to *postTriangular*() or *interpolation*(), q_{child} (the t-th waypoint of R), q_{parent}, and $q_{ancestor}$ are updated accordingly. If q_{parent} is the last point in R, f_{modify} is checked. Otherwise, the above process is repeated for the updated q_{child} and $q_{ancestor}$.

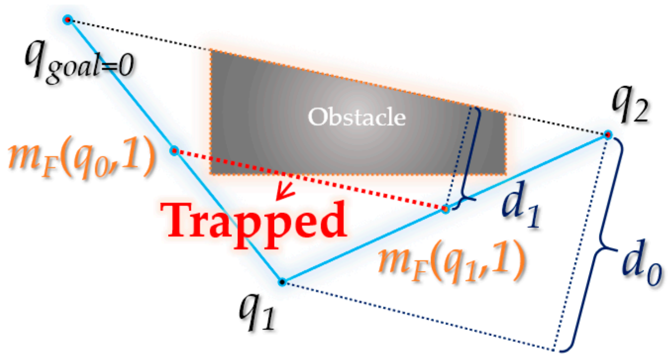
Here, path modification by *postTriangular*() deletes the waypoints and makes a more optimal path but has the effect of sharpening the path shape, and path modification by *interpolation*() creates a new waypoint between the waypoints. Adding/inserting has the effect of making a more optimal path while also smoothing the path shape. Of course, due to the effect of making a more optimal path, path modification by *postTriangular*() is more efficient than that by *interpolation*().

The input values of *postTriangular*() of the post triangular processing of the midpoint interpolation method consist of the path R, the focusing point index t, and the path modification f_{modify} from the post triangular processing of midpoint interpolation method.

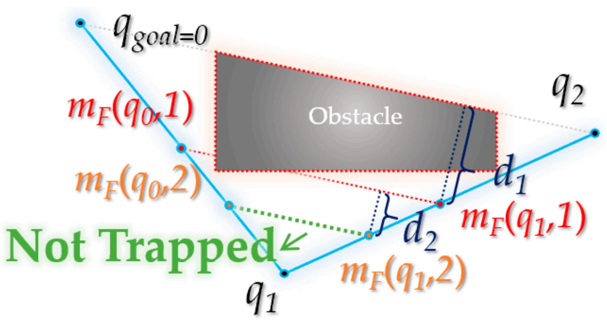
Rewiring is performed on the t-th waypoint q_{child} of R, the next point q_{parent}, and the next point $q_{ancestor}$ of that point again. First, the path between q_{child} and q_{parent} and the path between q_{parent} and $q_{ancestor}$ are deleted. Then the path is inserted between q_{child} and $q_{ancestor}$. Finally, f_{modify} returns "true" because the path has been modified. Here, Path <*A*, *B*> means a partial path from Waypoint A to Waypoint B in the complete path.

```
Algorithm 4 Pseudocode of Proposed Post Triangular Function
Input:
$R \leftarrow$ path R from $postTriProcOfMidInterpolation$
$t \leftarrow$ point index $t$ from $postTriProcOfMidInterpolation$
$f_{modify} \leftarrow$ boolean $f_{modify}$ from postTriProcOfMidInterpolation
Output:
$R \leftarrow$ modified path $R$
$f_{modify} \leftarrow$ result of boolean $f_{modify}$     //return by reference
Procedure postTriangular From postTriProcOfMidInterpolation
Begin
1        $q_{child} \leftarrow$ $t$-th point in $R$
2        $q_{parent} \leftarrow$ next point of $q_{child}$ in $R$
3        $q_{ancestor} \leftarrow$ next point of $q_{parent}$ in $R$
4        $R \leftarrow$ Delete path<$q_{child}$, $q_{parent}$> and path<$q_{parent}$, $q_{ancestor}$> from $R$
5        $R \leftarrow$ Insert path<$q_{child}$, $q_{ancestor}$> to $R$
6        $f_{modify} \leftarrow$ true
End
```

The goal of the proposed post triangular processing of midpoint interpolation method is to find an interpolation point ($m_F(q_0)$, $m_F(q_1)$) free from obstacle collisions between waypoints (q_0~q_1, q_1~q_2) while descending in the direction of the midpoint (q_1), as shown in Figure 6 in the interpolation process.



(**a**)



(**b**)

Figure 6. Details of post triangular processing of the midpoint interpolation method: (**a**) midpoint $m_F(q_0,1)$ of waypoint q_0, q_1 of path and midpoint $m_F(q_1,1)$ of q_1, q_2 are not free from obstacle collision; (**b**) midpoint $m_F(q_0,2)$ of interpolation point $m_F(q_0,1)$, q_1 and midpoint $m_F(q_1,2)$ between q_1, interpolation point $m_F(q_1,1)$ is free from obstacle collision.

However, the interpolation point m_F follows Equation (4):

$$m_F(q_i,k) = \begin{cases} \left(\frac{m_F(q_i,k-1).x+\xi(q_i).x}{2}, \frac{m_F(q_i,k-1).y+\xi(q_i).y}{2}\right), & k > 0 \\ q_i, & k = 0 \end{cases} \tag{4}$$

When the k-th interpolation point of the interpolation point q_i is $m_F(q_i,k)$, the 0-th interpolation point becomes itself q_i, and the first interpolation point is the midpoint of q_i and the point $\xi(q_i)$ next to q_i, and the second interpolation point becomes the midpoint of m_F $(q_i,1)$ and $\xi(q_i)$. That is, $m_F(q_i,k)$ $(k > 0)$ becomes the midpoint between $m_F(q_i,k-1)$ and $\xi(q_i)$. The following Algorithm 5 shows the pseudocode of *interpolation*() of the proposed post triangular processing of the midpoint interpolation method.

Algorithm 5 Pseudocode of Proposed Interpolation Function

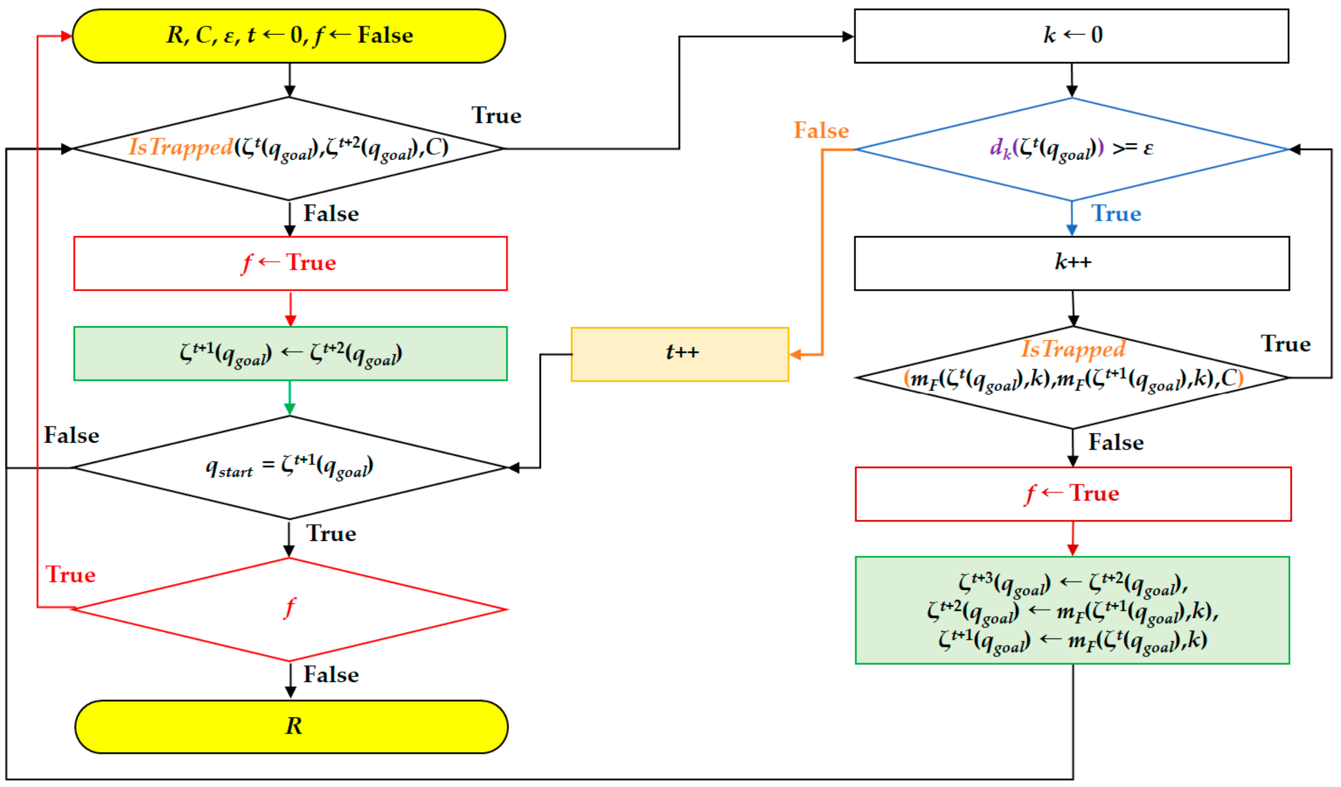
```
Input:
$R \leftarrow$ path R from postTriProcOfMidInterpolation
$C \leftarrow$ position set C from postTriProcOfMidInterpolation
$\varepsilon \leftarrow$ threshold value $\varepsilon$ from postTriProcOfMidInterpolation
$t \leftarrow$ point index t from postTriProcOfMidInterpolation
$f_{modify} \leftarrow$ boolean fmodify from postTriProcOfMidInterpolation
Output:
$R \leftarrow$ modified path $R$
$t \leftarrow$ updated point index $t$          // return by reference
$f_{modify} \leftarrow$ result of boolean fmodify          // return by reference
Initialize:
$q_{child} \leftarrow$ $t$-th point in $R$
$q_{parent} \leftarrow$ next point of $q_{child}$ in $R$
$q_{ancestor} \leftarrow$ next point of $q_{parent}$ in $R$
Procedure interpolation From postTriProcOfMidInterpolation
Begin
1     $d \leftarrow$ altitude of the triangle consisting of $q_{child}$, $q_{parent}$, and $q_{ancestor}$ with base <$q_{child}$, $q_{ancestor}$>
2     $m_a \leftarrow$ midpoint between $q_{child}$ and $q_{parent}$
3     $m_b \leftarrow$ midpoint between $q_{parent}$ and $q_{ancestor}$
4     While true Do
5         If $d >= \varepsilon$ Then
6             If not isTrapped($m_a$, $m_b$, $C$) Then
7                 $R \leftarrow$ Delete path<$q_{child}$, $q_{parent}$> and path<$q_{parent}$, $q_{ancestor}$> from R
8                 $R \leftarrow$ Insert path<$q_{child}$, $m_a$>, path<$m_a$, $m_b$>, and path<$m_b$, $q_{ancestor}$> to R
9                 $f_{modify} \leftarrow$ true
10                Break
11            Else
12                $d \leftarrow d$ / 2
13                $m_a \leftarrow$ midpoint between $m_a$ and $q_{parent}$
14                $m_b \leftarrow$ midpoint between $m_b$ and $q_{parent}$
15        Else
16            $t \leftarrow t + 1$
17            Break
End
```

The input values of *interpolation*() of the post triangular processing of midpoint interpolation method consist of the path R, the obstacle area information C, the focusing point index t, and the path modification f_{modify} from the post triangular processing of the midpoint interpolation method.

From the t-th waypoint q_{child} of R, the next point q_{parent}, and the next point $q_{ancestor}$ of that point, the height d of the triangle is obtained, m_a is the midpoint of q_{child} and q_{parent}, and m_b is the midpoint of q_{parent} and $q_{ancestor}$. If the path between m_a and m_b is free from obstacle collision (*isTrapped*()), the path between q_{child} and q_{parent} is deleted, and the path between q_{child} and m_a, the path between m_a and m_b, and the path between m_b and $q_{ancestor}$ are inserted. Moreover, since the path is modified, f_{modify} returns "true," and the method terminates. If the line segment between m_a and m_b is not free from obstacles, the value of d decreases by $1/2$, m_a is updated to the midpoint of m_a and q_{parent}, and m_b is updated to the midpoint of m_b and q_{parent}. It is then determined whether m_a and m_b are free from obstacles again. This repeated process proceeds until a case is found in which m_a and m_b are free from obstacles or d becomes smaller than ε. If d becomes smaller than ε, the value of t is increased by 1 and the method is terminated.

Figure 7 shows the overall flowchart of the proposed post triangular processing of the midpoint interpolation method. Here, $\xi^t(q_{goal})$ denotes the t-th next waypoint from the starting point q_{goal} of the path R, and $\xi^{t+n}(q_{goal})$ is the n-th next waypoint in the $\xi^t(q_{goal})$. That is, there are n waypoints between $\xi^t(q_{goal})$ and $\xi^{t+n}(q_{goal})$. In the flowchart shown in Figure 7, the stop condition follows the sequence:

1. Check whether the t-th ancestor node and the $(t+2)$-th ancestor node of q_{goal} are collision-free (Here, when t is 0, the 0-th ancestor means itself(q_{goal})).
2. If the result of Step 1 is not collision-free, compare the $d_k(\xi^t(q_{goal}))$ value of the t-th ancestor node of q_{goal} with ε.
3. If $d_k(\xi^t(q_{goal}))$ is less than ε, the value of t is incremented by 1, and if the parent node $(t+1)$ of the t-th ancestor node of q_{goal} after t is incremented is q_{start}, the algorithm is stopped (if f is False).



$R, C, \varepsilon, t \leftarrow 0, f \leftarrow$ False
$IsTrapped(\zeta^t(q_{goal}), \zeta^{t+2}(q_{goal}), C)$
True
False
$f \leftarrow$ True
$\zeta^{t+1}(q_{goal}) \leftarrow \zeta^{t+2}(q_{goal})$
False
$q_{start} = \zeta^{t+1}(q_{goal})$
True
True
f
False
R
$k \leftarrow 0$
False
$d_k(\zeta^t(q_{goal})) >= \varepsilon$
True
$k{+}{+}$
$t{+}{+}$
$IsTrapped(m_F(\zeta^t(q_{goal}),k), m_F(\zeta^{t+1}(q_{goal}),k), C)$
True
False
$f \leftarrow$ True
$\zeta^{t+3}(q_{goal}) \leftarrow \zeta^{t+2}(q_{goal})$,
$\zeta^{t+2}(q_{goal}) \leftarrow m_F(\zeta^{t+1}(q_{goal}),k)$,
$\zeta^{t+1}(q_{goal}) \leftarrow m_F(\zeta^{t}(q_{goal}),k)$

Figure 7. Flowchart of post triangular processing of midpoint interpolation method.

The stopping criterion of the proposed algorithm is based on ε. As shown in Figures 4 and 7, when the value of $d_k(\xi^t(q_{goal}))$ becomes smaller than the ε, the algorithm stops the loop. As Equations (1) and (2) and Figure 6 show, the value $d_k(\xi^t(q_{goal}))$ decreases deterministically.

4. Experimental Results

The path between the RRT in various environment maps through simulation and the RRT algorithm to which the proposed post triangular processing of the midpoint interpolation method is applied were used to validate the performance of the method proposed in this paper, and the path planning results were compared.

The performance measures compared were the average values after repeating the trial 100 times (sampling position was changed for each trial) of the path length (px) and the planning time (ms) of the first complete path (the first complete path to reach a destination point from a starting point).

Various environment maps have been examined and used to validate the performance of the proposed path planning algorithms in related works. Since the efficiency of the performance measures expected during the experiment varies somewhat based on the composition of obstacles (e.g., number, location, shape), it is important to choose which environment map to utilize carefully.

The four environment maps used in the experiment are shown in Figure 8. The four environment maps were created by partially referring to the experimental environment proposed by Han in 2017 [21]. All environment maps were 600 × 600 px in size, with a 30 px step length. The start points (S) are represented by green circles, while the destination points (G) are represented by purple circles. Obstacles are black polygons with yellow contours (blue in the experimental results).

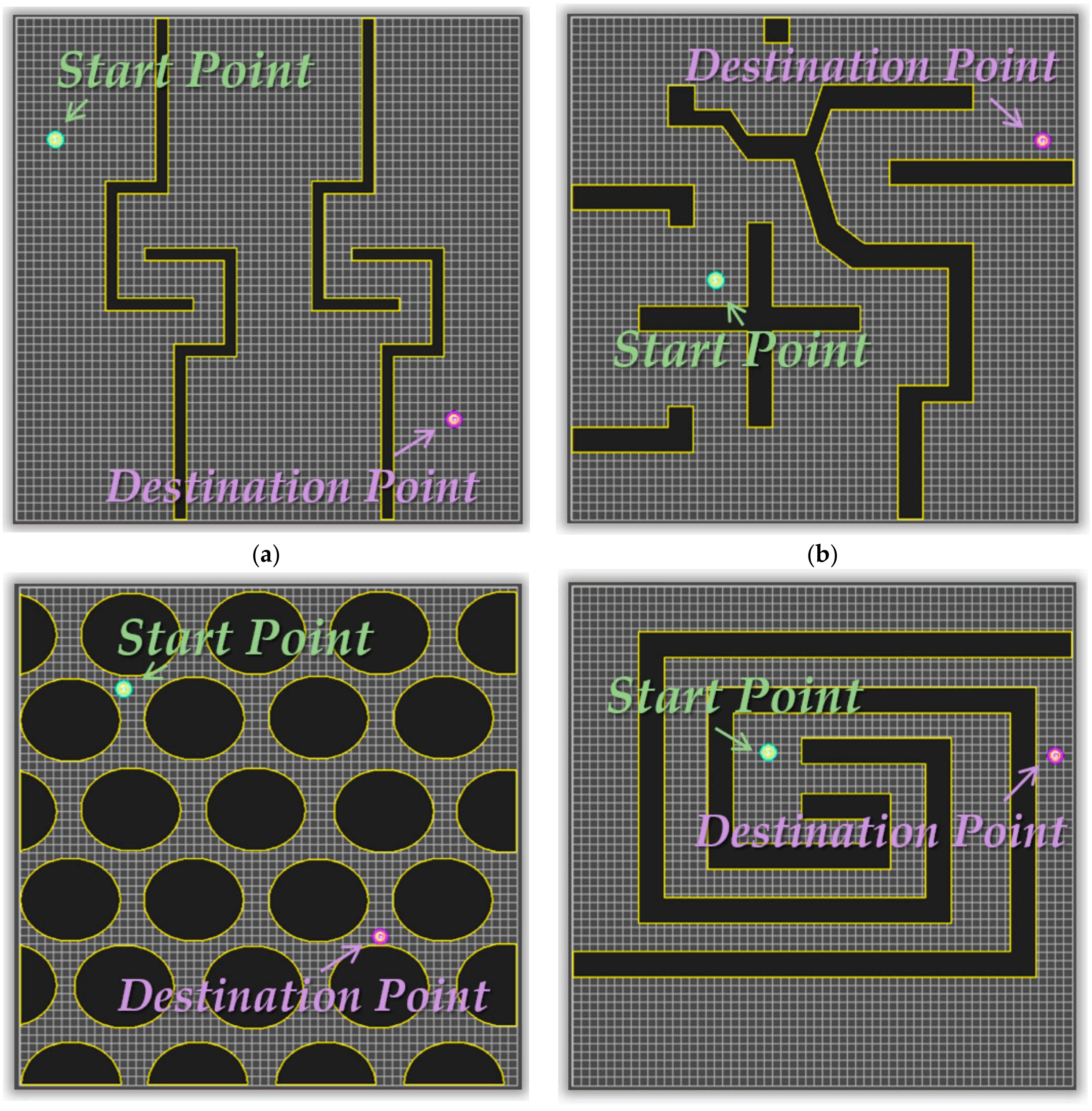


Figure 8. Environment maps for experiments: (**a**) Map 1; (**b**) Map 2; (**c**) Map 3; (**d**) Map 4.

Map 1 of Figure 8a appears to be an efficient environment for validating optimality and completeness but a weak environment for sampling-based path planning algorithms like the RRT algorithm. Many samplings are required since the probability of finding a solution is low. Map 2 of Figure 8b appears to be an efficient environment for validating the optimality and completeness of the path planning algorithm. Map 3 of Figure 8c is an environment that is efficient for validating the optimality and the planning time of the path planning algorithm, as it consists of obstacles (50 vertices) that approximate circles. Map 4 of Figure 8d is an environment that is efficient for validating the optimality and the planning time of the path planning algorithm but a weak environment for sampling-based path planning algorithms such as the RRT algorithm.

The number of samples and planning time required increase drastically when the path to the destination point is narrow or there are few entrances since the sampling-based path planning algorithm depends on probabilistic completeness.

The specification of the computer used in the simulation is shown in Table 1. The simulator used for the simulation [8] was developed based on C# WPF (Microsoft Visual Studio Community 2019 Version 16.1.6 Microsoft .NET Framework Version 4.8.03752), and only a single thread was used for calculations except for the visual part. Generally, depending on the specification of the computer, the result of performance measurements, such as planning time, may vary during the simulation.

Table 1. Computer performance for simulation.

H/W	Specification
CPU	Intel Core i7-6700k 4.00 GHz (8 CPUs)
RAM	32,768 MB (32 GB DDR4)

We validate the experimental results (path length, planning time) in the four environment maps in which the post triangular processing of the midpoint interpolation method (ε: 50, 30, 10 px) is applied to the RRT algorithm and its path planning results. Since ε requires a higher amount of computation as it gets smaller, it was set to a value close to the 30 px step length of the experimental environment.

The experimental results for each map consist of a figure and table. The figure is a path planning result for each algorithm shown in the case of a single trial (the figure for each algorithm is not the result of repeated trials). The table shows an average value of the results of planning time and path length from the path planning repeated trials. There may be a significant difference between the performance observed visually and the numerical results in the table for one of the repeated trials. The form of the planned path for each algorithm is shown in the figure for visual reference. Furthermore, the proposed post triangular processing of the midpoint interpolation method is used to see whether there is a region where the piecewise linear path is smoothed.

The path planning results are shown in Figure 9 for Map 1 among the specified environment maps for each algorithm. In terms of appearance, the one to which the post triangular processing of the midpoint interpolation method (ε: 10 px) was applied seems to have a smoother slope compared to those of the other algorithms, and the path length with the method (ε: 10 px) seems to be the shortest.

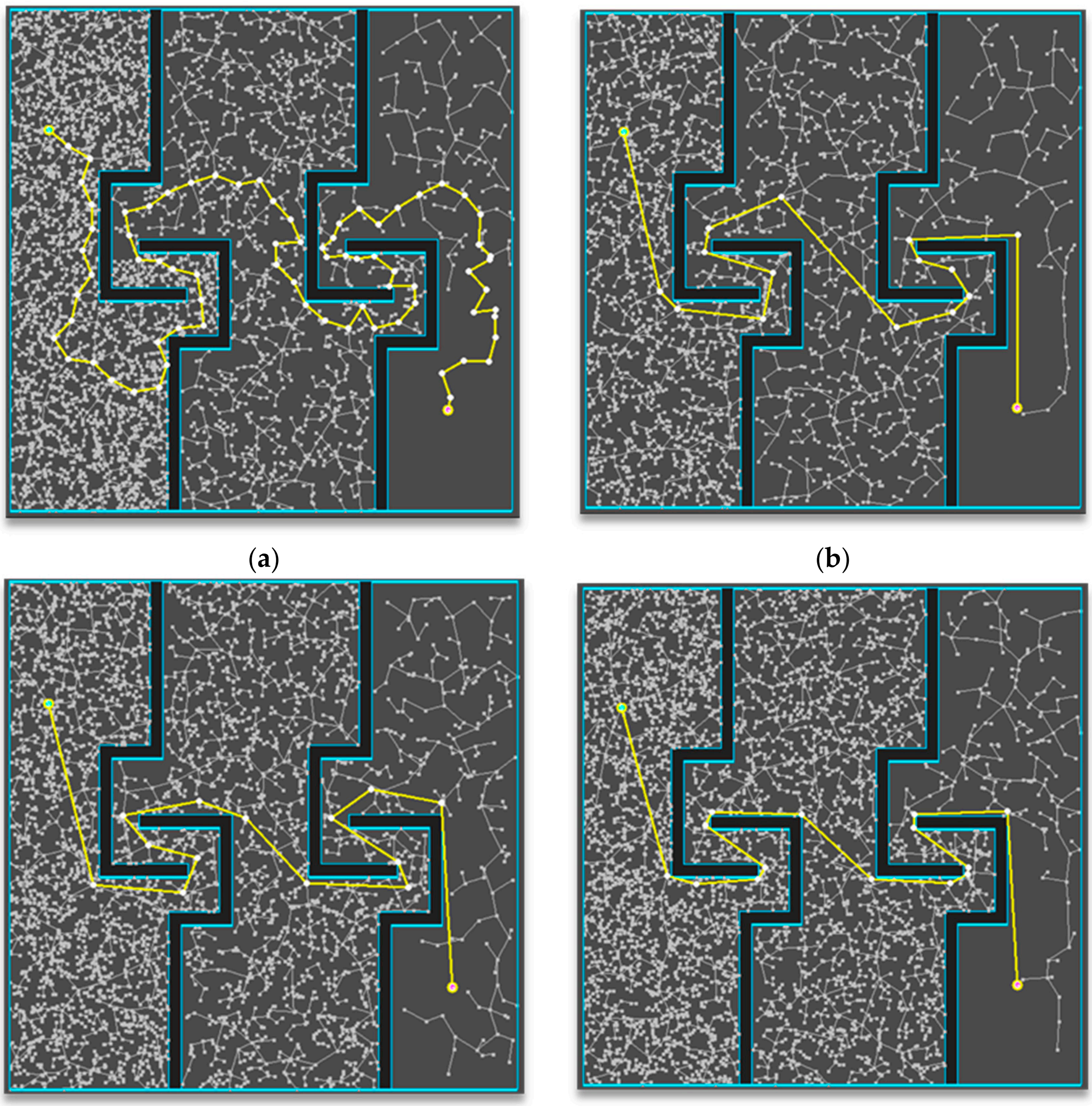

(**a**) (**b**) (**c**) (**d**)

Figure 9. Experimental results of Map 1: (**a**) RRT; (**b**) proposed method (ε: 50 px) applied; (**c**) proposed method (ε: 30 px) applied; (**d**) proposed method (ε: 10 px) applied.

The path planning results (average values after repeating the trial 100 times) are shown in Table 2 for Map 1 among the specified environment maps for each algorithm. When applying the post triangular processing of the midpoint interpolation method (ε: 10 px), the path length becomes 62% (1224/1994(%)) compared to the RRT, which is the shortest of all the algorithms, and the planning times are all similar.

Table 2. Experimental results of Map 1 (numbers in parentheses (averages of repeating trial 100 times) are relative ratios to RRT results (values less than 1 are counted as 1)).

Performance	**RRT**	**Proposed Method (ε: 50 px)**	**Proposed Method (ε: 30 px)**	**Proposed Method (ε: 10 px)**
Path length (px)	1944 (100%)	1403 (72%)	1325 (68%)	1224 (62%)
Planning time (ms)	697 (100%)	698 (100%)	698 (100%)	698 (100%)

The path planning results are shown in Figure 10 for Map 2 among the specified environment maps for each algorithm. In terms of appearance, the one to which the post triangular processing of midpoint interpolation method (ε: 10 px) was applied seems to have a smoother slope compared to those of the other algorithms, and the path length with the method (ε: 10 px) seems to be the shortest.

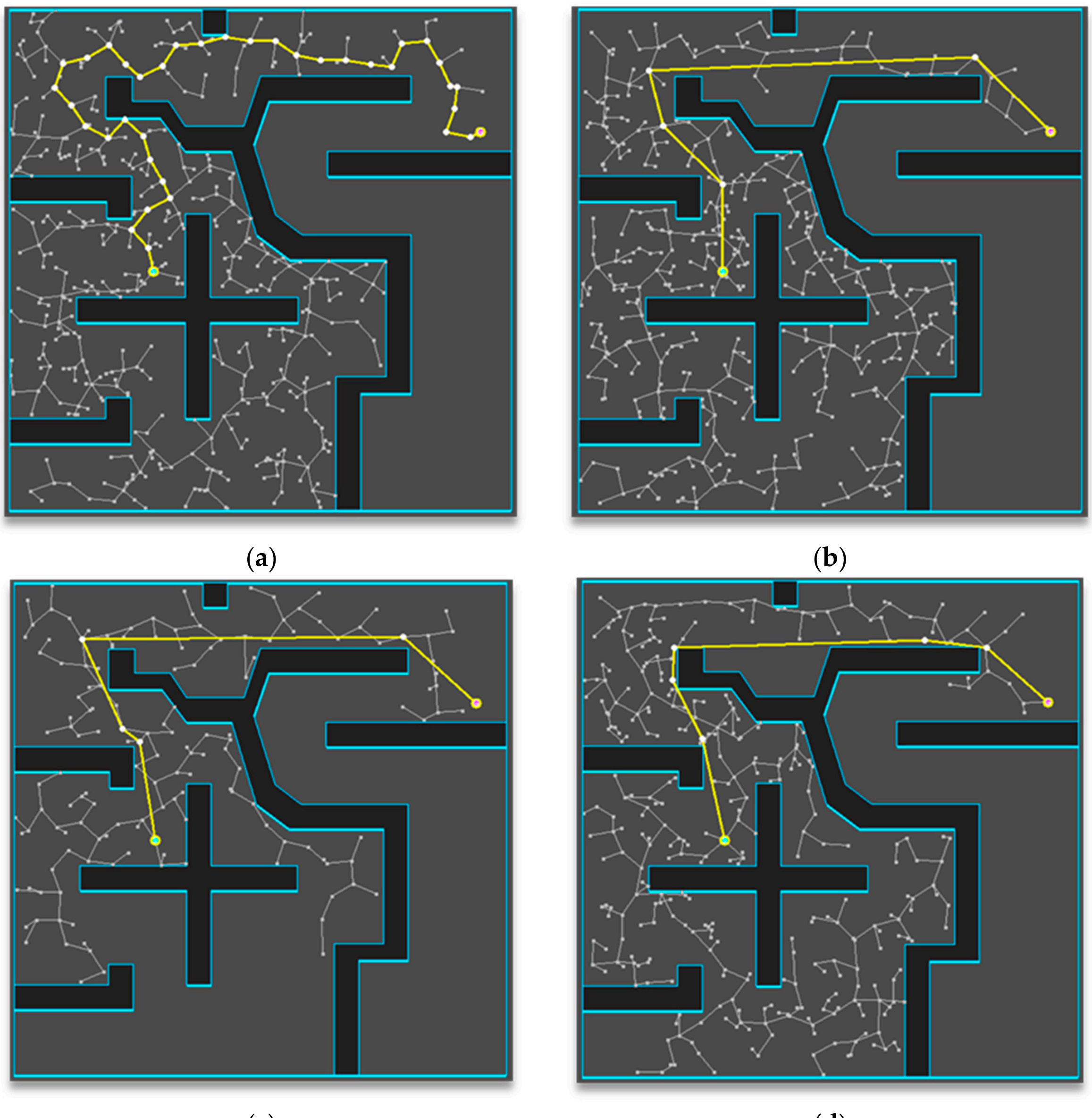


Figure 10. Experimental results of Map 2: (**a**) RRT; (**b**) Proposed method (ε: 50 px) applied; (**c**) Proposed method (ε: 30 px) applied; (**d**) Proposed method (ε: 10 px) applied.

The path planning results (average values after repeating the trial 100 times) are shown in Table 3 for Map 2 among the specified environment maps for each algorithm. By applying the post triangular processing of midpoint interpolation method (ε: 10 px), the path length becomes 74% (730/986(%)) compared to the RRT, which is the shortest of all the algorithms, and the planning time is similar to that of the RRT algorithm when the method (ε: 50 px) is applied. However, the absolute time difference is 1 ms, which seems to be large when the method is applied because it is an environment that requires less planning time.

Table 3. Experimental results of Map 2 (numbers in parentheses (averages of repeating trial 100 times) are relative ratios to RRT results (values less than 1 are counted as 1)).

Performance	RRT	Proposed Method (ε: 50 px)	Proposed Method (ε: 30 px)	Proposed Method (ε: 10 px)
Path length (px)	986 (100%)	780 (79%)	752 (76%)	730 (74%)
Planning time (ms)	10 (100%)	10 (100%)	11 (110%)	11 (110%)

The path planning results are shown in Figure 11 for Map 3 among the specified environment maps for each algorithm. In terms of appearance, the one to which the post triangular processing of midpoint interpolation method (ε: 10 px) was applied seems to have a smoother slope compared to those of the other algorithms, and the path length with the method (ε: 10 px) seems to be the shortest.

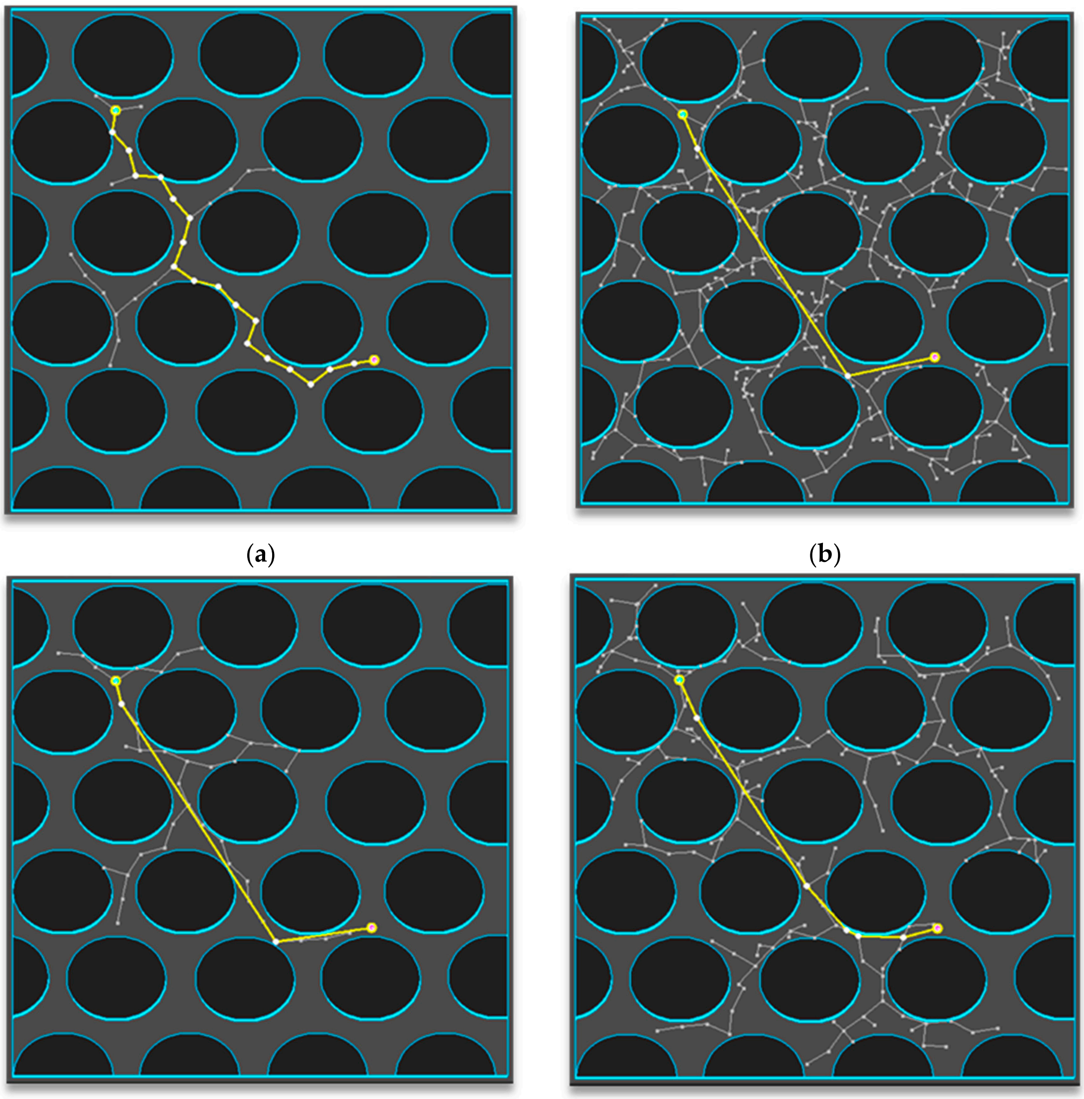

(**a**) (**b**) (**c**) (**d**)

Figure 11. Experimental results of Map 3: (**a**) RRT; (**b**) Proposed method (ε: 50 px) applied; (**c**) Proposed method (ε: 30 px) applied; (**d**) Proposed method (ε: 10 px) applied.

The path planning results (average values after repeating the trial 100 times) are shown in Table 4 for Map 3 among the specified environment maps for each algorithm. By applying the post triangular processing of midpoint interpolation method (ε: 10 px), the path length becomes 82% (505/613(%)) compared to the RRT, which is the shortest of all the algorithms, and the planning time is the most similar to that of the method (ε: 10 px), as it takes 16% more time than the RRT algorithm. However, the absolute time difference is 2 ms, and since it is an environment that requires less planning time, the difference appears to be large when the method is applied.

Table 4. Experimental results of Map 3 (numbers in parentheses (averages of repeating 100 times) are relative ratios to the RRT results (values less than 1 are counted as 1)).

Performance	RRT	Proposed Method (ε: 50 px)	Proposed Method (ε: 30 px)	Proposed Method (ε: 10 px)
Path length (px)	613 (100%)	535 (87%)	512 (83%)	505 (82%)
Planning time (ms)	6 (100%)	7 (116%)	8 (133%)	8 (133%)

The path planning results are shown in Figure 12 for Map 4 among the specified environment maps for each algorithm. In terms of appearance, the one to which the post triangular processing of the midpoint interpolation method (ε: 30 px) was applied seems to have a smoother slope compared to those of the other algorithms, and the path length with the method (ε: 10 px) seems to be the shortest.

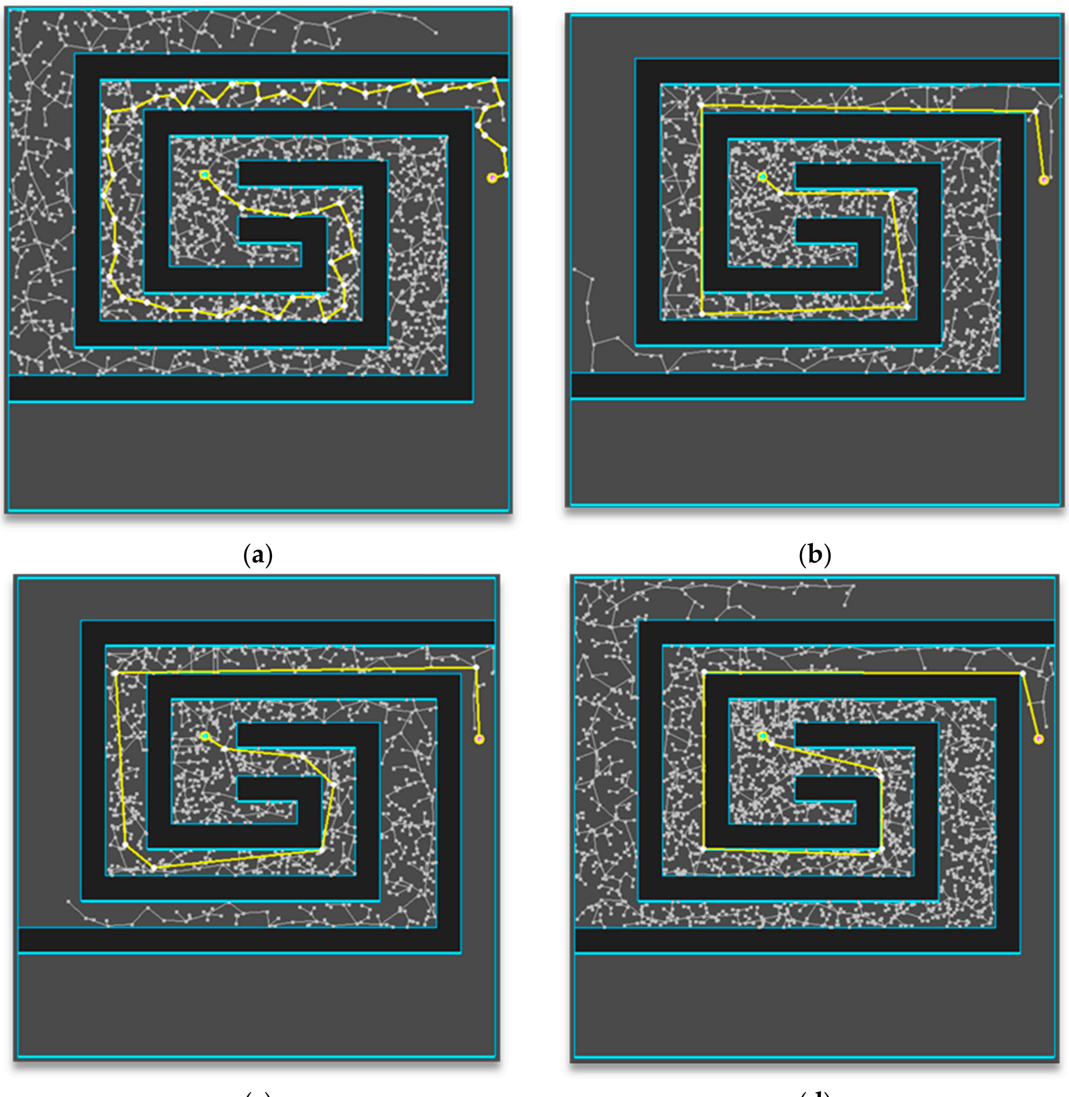

(**a**) (**b**) (**c**) (**d**)

Figure 12. Experimental results of Map 4: (**a**) RRT; (**b**) Proposed method (ε: 50 px) applied; (**c**) Proposed method (ε: 30 px) applied; (**d**) Proposed method (ε: 10 px) applied.

The path planning results (average values after repeating the trial 100 times) are shown in Table 5 for Map 4 among the specified environment maps for each algorithm. By applying the post triangular processing of the midpoint interpolation method (ε: 10 px), the path length becomes 77% (1190/1536(%)) compared to the RRT, which is the shortest of all the algorithms, and all the planning times are similar.

Table 5. Experimental results of Map 4 (numbers in parentheses to (averages of repeating 100 times) are relative ratios to RRT results (values less than 1 are counted as 1)).

Performance	RRT	Proposed Method (ε: 50 px)	Proposed Method (ε: 30 px)	Proposed Method (ε: 10 px)
Path length (px)	1536 (100%)	1284 (83%)	1261 (82%)	1190 (77%)
Planning time (ms)	1752 (100%)	1753 (100%)	1753 (100%)	1753 (100%)

Consequently, the post triangular processing of the midpoint interpolation method (ε: 10 px) performed well in the path length aspect for all maps, demonstrating that the proposed method is efficient in terms of optimality.

5. Conclusions

In this research, the post triangular processing of the midpoint interpolation method minimized the planning time and shortened the path length of the sampling-based algorithm.

The proposed post triangular processing of the midpoint interpolation method was closer to the optimal path and somewhat solved the sharp path problem through the interpolation process. Furthermore, all path planning algorithms that plan a locally piecewise linear path could apply the proposed algorithm. This strength has significance in that it can be applied not only to the algorithm presented in this paper but also to various path planning algorithms. We validated the performance of the proposed method after it was applied to the RRT algorithm and its path planning results through simulation. It was verified that the path length was shortened by 18–38% (average 26%) depending on the threshold ε when applied to the RRT algorithm in the four different environment maps. As a result, the RRT algorithm applying the proposed post triangular processing of the midpoint interpolation method showed a more optimal path.

The proposed post triangular processing of the midpoint interpolation method is based on a general global planning case in which a robot first discovers a global path from its start point to its destination point before beginning to navigate. In dynamic environments, not only local planners but also global planners must deal with kinodynamic problems in real-time. Furthermore, the method proposed in this paper is more efficient in terms of the optimality of robot path planning, but optimality is not guaranteed. Future work should determine how to make a path that is closer to the optimal path.

Author Contributions: Idea and conceptualization: J.-G.K. and J.-W.J.; Methodology: J.-G.K. and J.-W.J.; Software: J.-G.K. and J.-W.J.; Experiment: J.-G.K. and J.-W.J.; Validation: J.-G.K., Y.-S.C. and J.-W.J.; Investigation: J.-G.K. and J.-W.J.; Resources: J.-G.K. and J.-W.J.; Writing: J.-G.K., Y.-S.C. and J.-W.J.; Visualization: J.-G.K. and J.-W.J.; Project administration: J.-W.J. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported by a National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIT) (No. 2020R1F1A1074974) and by the Ministry of Trade, Industry, and Energy (MOTIE) and the Korea Institute for Advancement of Technology (KIAT) through the International Cooperative R&D program (Project No. P0016096). It was also supported by the Technology development Program(S3041234) funded by the Ministry of SMEs and Startups (MSS, Korea) and by the MSIT (Ministry of Science and ICT), Korea, under the ITRC (Information Technology Research Center) support program (IITP-2021-2020-0-01789) supervised by the IITP (Institute for Information & Communications Technology Planning & Evaluation).

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

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

References

1. Lindner, L.; Sergiyenko, O.; Moises, R.L.; Ivanov, M.; Julio, C.R.Q.; Daniel, H.B.; Wendy, F.F.; Vera, T.; Fabian, N.M.R.; Mercorelli, P. Machine Vision System Errors for Unmanned Aerial Vehicle Navigation. In Proceedings of the IEEE International Symposium on Industrial Electronics, Edinburgh, UK, 19–21 June 2017; pp. 1615–1620.
2. Sergiyenko, O.; Wendy, F.F.; Mercorelli, P. *Machine Vision and Navigation*, 1st ed.; Springer: Cham, Switzerland, 2020.
3. Abdi, A.; Adhikari, D.; Park, J.H. A novel hybrid path planning method based on q-learning and neural network for robot arm. *Appl. Sci.* **2021**, *11*, 6770. [CrossRef]
4. Schwab, K. *The Fourth Industrial Revolution*, 1st ed.; Crown Business: New York, NY, USA, 2017.
5. Wang, X.; Luo, X.; Han, B.; Chen, Y.; Liang, G.; Zheng, K. Collision-free path planning method for robots based on an improved rapidly-exploring random tree algorithm. *Appl. Sci.* **2020**, *10*, 1381. [CrossRef]
6. LaValle, S.M.; Kuffner, J.J., Jr. Randomized kinodynamic planning. *Int. J. Robot. Res.* **2001**, *20*, 378–400. [CrossRef]
7. Kuffner, J.J., Jr.; LaValle, S.M. RRT-Connect: An Efficient Approach to Single-Query Path Planning. In Proceedings of the IEEE International Conference on Robotics and Automation, San Francisco, CA, USA, 24–28 April 2000; Volume 2, pp. 995–1001.
8. Kang, J.-G.; Lim, D.-W.; Choi, Y.-S.; Jang, W.-J.; Jung, J.-W. Improved RRT-connect algorithm based on triangular inequality for robot path planning. *Sensors* **2021**, *21*, 333–364. [CrossRef] [PubMed]
9. Roy, D. Visibility graph based spatial path planning of robots using configuration space algorithms. *Robot. Auton. Syst.* **2009**, *24*, 1–9. [CrossRef]
10. Katevas, N.I.; Tzafestas, S.G.; Pnevmatikatos, C.G. The approximate cell decomposition with local node refinement global path planning algorithm: Path nodes refinement and curve parametric interpolation. *J. Intell. Robot. Syst.* **1998**, *22*, 289–314. [CrossRef]
11. Warren, C.W. Global Path Planning using Artificial Potential Fields. In Proceedings of the International Conference on Robotics and Automation, Scottsdale, AZ, USA, 14–19 May 1989; Volume 1, pp. 316–321.
12. LaValle, S.M. *Rapidly-Exploring Random Trees: A New Tool for Path Planning*; Springer: London, UK, 1998.
13. Karaman, S.; Frazzoli, E. Sampling based algorithms for optimal motion planning. *Int. J. Robot. Res.* **2011**, *30*, 846–894. [CrossRef]
14. Gammell, J.D.; Srinivasa, S.S.; Barfoot, T.D. Informed RRT*: Optimal Sampling based Path Planning Focused via Direct Sampling of an Admissible Ellipsoidal Heuristic. In Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems, Chicago, IL, USA, 14–18 September 2014; pp. 2997–3004.
15. Klemm, S.; Oberländer, J.; Hermann, A.; Roennau, A.; Schamm, T.; Zollner, J.M.; Dillmann, R. RRT*-Connect: Faster, Asymptotically Optimal Motion Planning. In Proceedings of the IEEE International Conference on Robotics and Biomimetics, Zhuhai, China, 6–9 December 2015; pp. 1670–1677.
16. Islam, F.; Nasir, J.; Malik, U.; Ayaz, Y.; Hasan, O. Rrt*-smart: Rapid Convergence Implementation of rrt* towards Optimal Solution. In Proceedings of the IEEE International Conference on Mechatronics and Automation, Chengdu, China, 5–8 August 2012; pp. 1651–1656.
17. Jeong, I.-B.; Lee, S.-J.; Kim, J.-H. Quick-RRT*: Triangular inequality based implementation of RRT* with improved initial solution and convergence rate. *Expert Syst. Appl.* **2019**, *123*, 82–90. [CrossRef]
18. Jung, J.-W.; So, B.-C.; Kang, J.-G.; Lim, D.-W.; Son, Y. Expanded Douglas–Peucker polygonal approximation and opposite angle based exact cell decomposition for path planning with curvilinear obstacles. *Appl. Sci.* **2019**, *9*, 638. [CrossRef]
19. Jung, J.-W.; So, B.-C.; Kang, J.-G.; Jang, W.-J. Circumscribed Douglas-Peucker Polygonal Approximation for Curvilinear Obstacle Representation. In Proceedings of the IEEE 2019 7th International Conference on Robot Intelligence Technology and Applications (RiTA), Daejeon, Korea, 1–3 November 2019; pp. 237–241.
20. Kwon, J.; Choi, K. Kinodynamic model identification: A unified geometric approach. *IEEE Trans. Robot.* **2021**, *37*, 1100–1114. [CrossRef]
21. Han, J. Mobile robot path planning with surrounding point set and path improvement. *Appl. Soft Comput.* **2017**, *57*, 35–47. [CrossRef]