sensors

MDPI

Article

A Bidirectional Interpolation Method for Post-Processing in Sampling-Based Robot Path Planning

Tae-Won Kang [1], Jin-Gu Kang [2] and Jin-Woo Jung [2,*]

[1] Department of Artificial Intelligence, Dongguk University, Seoul 04620, Korea; ktw3388@dgu.ac.kr
[2] Department of Computer Science and Engineering, Dongguk University, Seoul 04620, Korea; kanggu12@dongguk.edu
* Correspondence: jwjung@dongguk.edu; Tel.: +82-2-2260-3812

Abstract: This paper proposes a post-processing method called bidirectional interpolation method for sampling-based path planning algorithms, such as rapidly-exploring random tree (RRT). The proposed algorithm applies interpolation to the path generated by the sampling-based path planning algorithm. In this study, the proposed algorithm is applied to the path created by RRT-connect and six environmental maps were used for the verification. It was visually and quantitatively confirmed that, in all maps, not only path lengths but also the piecewise linear shape were decreased compared to the path generated by RRT-connect. To check the proposed algorithm's performance, visibility graph, RRT-connect algorithm, Triangular-RRT-connect algorithm and post triangular processing of midpoint interpolation (PTPMI) were compared in various environmental maps through simulation. Based on these experimental results, the proposed algorithm shows similar planning time but shorter path length than previous RRT-like algorithms as well as RRT-like algorithms with PTPMI having a similar number of samples.

Keywords: bidirectional interpolation method; post-processing; RRT-connect; triangular RRT-connect; midpoint interpolation; sampling-based path planning

Citation: Kang, T.-W.; Kang, J.-G.; Jung, J.-W. A Bidirectional Interpolation Method for Post-Processing in Sampling-Based Robot Path Planning. *Sensors* **2021**, *21*, 7425. https://doi.org/10.3390/s21217425

Academic Editor: Andrzej Stateczny

Received: 29 September 2021
Accepted: 3 November 2021
Published: 8 November 2021

1. Introduction

This study deals with the path planning of a mobile robot [1]. Strictly speaking, path planning can be divided into global planning on an entire map and local planning on a portion of the map [2]. In this paper, path planning refers to global planning.

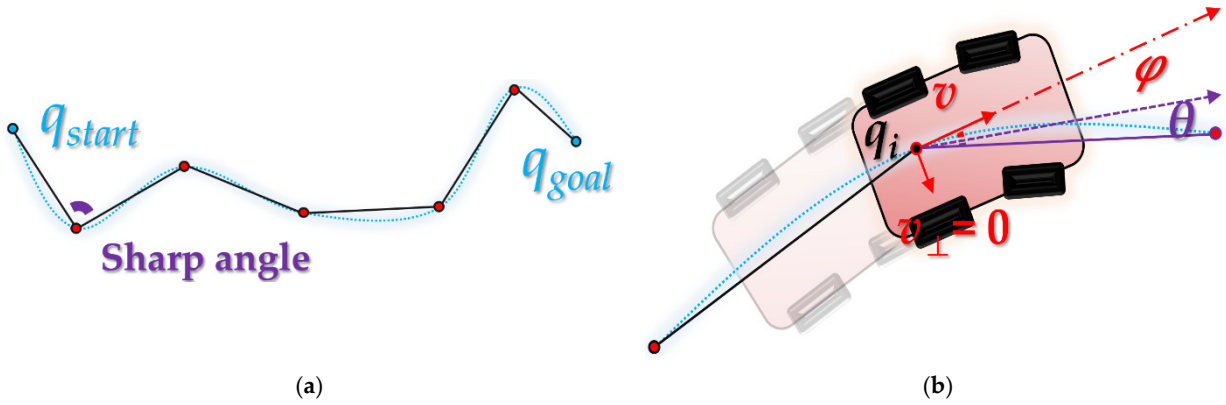
Path planning involves plotting a path that a mobile robot can follow to efficiently move from a starting point to a goal point in Euclidean space, avoiding obstacles, with respect to optimality, clearance and completeness [3]. Optimality refers to always being able to plan a path with the optimal path length. Clearance refers to how low the probability is that the mobile robot will collide with an obstacle. Completeness indicates that a path can always be planned in an environment in which a solution exists.

This paper mainly deals with the sampling-based rapidly-exploring random tree (RRT)-like algorithm [4]. The RRT-like algorithms are being applied in various ways. For example, there is a method that generates an optimal path by applying a triangular inequality [5] and a method applicable to kinodynamic planning [6]. The RRT algorithm can be summarized as a method of planning a path by repeatedly inserting a randomly sampled location as a child node in a tree with the starting point as the root node until reaching the goal point. In this method, the tree trunk extends in the shape of a stochastic fractal and attempts to reach the goal.

Sampling-based algorithms, including RRT algorithm, have the advantage of being able to plan a path with fewer computations than classical path planning algorithms such as those for visibility-graph-based [7], cell-decomposition-based [8] and potential-field-based [9] methods. However, RRT does not guarantee optimality and probabilistic completeness [4]. Probabilistic completeness means that if the number of sampling nodes

is unlimited, completeness is guaranteed, but if the number of sampling nodes is limited, completeness is not guaranteed.

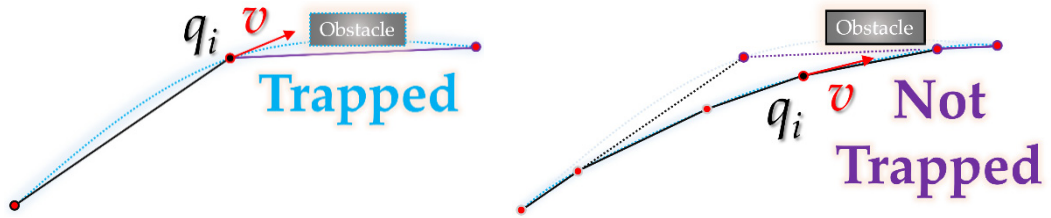
The purpose of this study is to guarantee completeness, improve optimality and improve collision-avoidance of RRT-like algorithms for path planning. When a path is planned by the RRT algorithm, it has a stochastic fractal shape, and locally it tends to have a piecewise linear shape [10] as shown in Figure 1a. The piecewise linear contour can result in collisions [11] owing to the kinematic constraints [12] of the mobile robot, as shown in Figure 1b.



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

Figure 1. Path of RRT-like algorithms with sharp angle: (**a**) piecewise linear path; (**b**) situation with turning radius φ as per the kinematic constraints of mobile robot with velocity v at node (turn penalty θ).

When the mobile robot moves along the planned path or local planning [13] path, it may collide with an obstacle as shown in Figure 2a because of kinematic constraints on the path with sharp angles. As this leads to serious concerns from the perspective of clearance, path planning must also consider kinodynamic planning [6,14,15]. In particular, as there are fewer waypoints on the planned path (a tree node in the RRT algorithm) or, alternatively, the distance between waypoints is high, kinematic error is more likely to occur. This may be exacerbated when the mobile robot travels at high speed and the control error increases.



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

Figure 2. Path needing kinodynamic planning (when a mobile robot with kinematic constraints moves at a velocity v at an arbitral point q_i): (**a**) situation in which the robot collides with the obstacle; (**b**) situation in which the obstacle is avoided by increasing way points.

In this study, given that the RRT algorithm does not guarantee optimality, we aim to generate a path that is closer to the optimum. Simultaneously, the scope of the study includes solving the local clearance problem of the stochastic fractal-shaped path.

However, when the path is first plotted from the starting point to the goal point, only the first complete path is dealt with. That is, the complete path or convergence path that is closer to the optimum through additional sampling after the initial complete path is not considered within the scope of the current study.

The RRT* algorithm, an improved version of the RRT algorithm, further optimizes the convergence path, wherein the convergence rate of the first complete path is higher

than that of the RRT algorithm. Consequently, it is possible to plan a path closer to the optimum. However, the amount of computation required to arrive at the convergence path is very high [16]. Recently, the post triangular processing of midpoint interpolation (PTPMI) method has been proposed to solve this problem [17]. Therefore, in this study, we prioritize finding a solution, which is the advantage of the sampling-based algorithm. In addition, the purpose of this study is to generate a path that is closer to the optimum for a similar computation time without significantly increasing the amount of computation compared to related studies.

We propose the bidirectional interpolation method for post-processing, which allows the RRT algorithm to plan a path that is closer to optimal and to mitigate the local clearance problem.

Visibility-graph-based path planning creates an optimal path. However, in order to make this path clearer in a real environment, it is necessary to use a sensor device with high resolution. The proposed method in this paper can generate a path that is close to optimal even in sensor equipment with low resolution. Therefore, the cost for the sensor could be saved.

The overall structure of this paper is as follows. Section 2 deals with related works. Section 2.1 deals with classical path planning and Section 2.2 deals with the sampling-based path planning algorithm. Section 3 deals with the proposed method (bidirectional interpolation method for post-processing). Section 4 deals with experimental results in which the path lengths and planning times are compared and analyzed through simulation to verify the performance of the proposed method.

2. Related Works

2.1. Classical Path Planning

2.1.1. Visibility Graph

The visibility graph algorithm guarantees optimality and completeness, but the clearance is very low. The cell decomposition algorithm guarantees completeness and high clearance, but does not guarantee optimality. The potential field algorithm has very high clearance, but cannot guarantee completeness and optimality by local minima [9].

In particular, as the visibility graph algorithm guarantees optimality in 2D configuration space [18], it is often used as a comparative experiment target for path planning method studies. We also compare the RRT algorithm post-processed using the proposed method to the visibility graph in terms of path length, in order to gauge their relative optimality.

The visibility graph algorithm was proposed by Tomas Lozano-Perez in 1979. As shown in Figure 3a, the start point, goal point and the vertices of all obstacle polygons are connected in a graph form. As shown in Figure 3b, it guarantees optimality by searching for the shortest path from the node at the starting point to the node at the goal. At this time, the case with an edge connecting a node and a node passing through an obstacle is excluded.

2.1.2. Limitation of Classical Path Planning

All the classical path planning methods, including the visibility graph algorithm, have their own advantages and disadvantages, but a common limitation is that they are difficult to apply to a dynamic environment because they involve a large amount of computation.

The path planning algorithms of the classical method introduced above plan the path using obstacle area information, unlike the sampling-based algorithm. As the visibility graph algorithm and the cell decomposition algorithm plan a path using the vertices of the obstacle polygon, the amount of computation becomes very large when the number of vertices of the entire obstacle is large. In addition, given that the visibility graph algorithm and cell decomposition algorithm can be applied only when the shape of the obstacle is polygonal, polygon approximation is required when a curved obstacle is given as an input value in a vector map rather than a grid map [19,20].

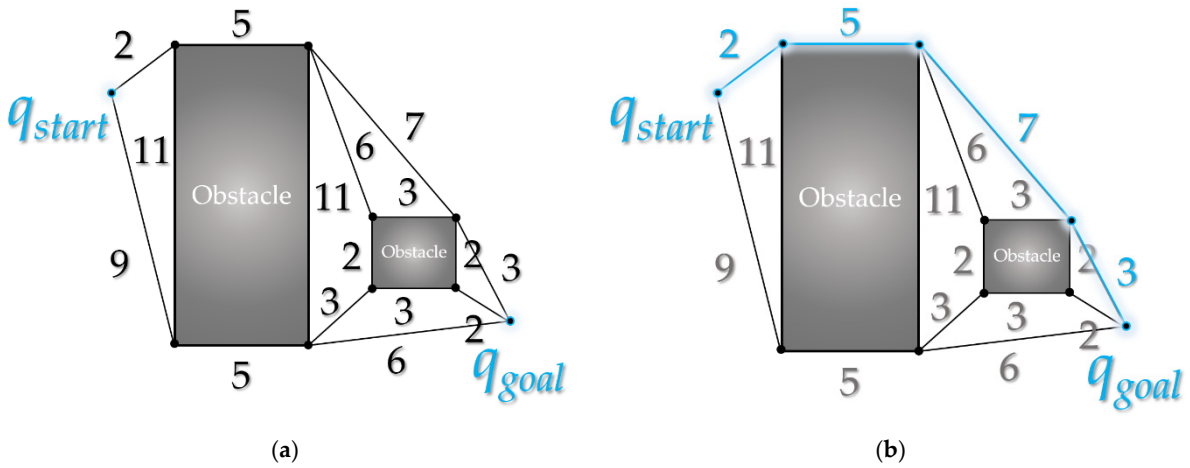


Figure 3. Overview of visibility graph: (**a**) graph (the number on the edge is the length of that edge) connecting start point q_{start}, goal point q_{goal} and the vertices of all obstacle polygons; (**b**) shortest path from start point q_{start} node to goal point q_{goal} node.

The precision of modern mobile robot sensing technology and equipment (such as radar and LiDAR) is very high, so obstacles can be mapped with high resolution (the number of polygonal vertices of obstacles is proportional to the sensing precision) [21]. Therefore, path planning through classical methods such as visibility graph algorithm is difficult to apply to dynamic environments. One solution is to reduce the computational amount of the classical method by reducing the number of obstacle polygon vertices through the polygon approximation algorithm. However, this distorts the actual shape, and therefore unintentional merging between polygons or obstacle collision problems caused by distorted areas can cause fatal problems in route planning.

Unlike the classical method, the sampling-based method does not actively use the obstacle area information for path planning; it is only used for obstacle collision inspection. Therefore, the amount of computation does not increase significantly depending on the type of obstacle. In other words, it is suitable as a modern mobile robot route planning method because it can plan a route in a short time even in a dynamic environment mapped at a high resolution.

2.2. Sampling-Based Path Planning Algorithms

Sampling-based path planning methods include the RRT algorithm and the probabilistic roadmap method (PRM) algorithm [22]. Well-known improvements to the RRT algorithm include RRT-connect [23], RRT* [14], informed RRT [24] and quick*-RRT [10] algorithms. This section deals only with RRT and RRT-connect algorithms.

As the purpose of this paper is specifically to improve the performance of the first complete path of the RRT algorithm, research on the efficient convergence path or the RRT* algorithm for improving convergence speed or the informed RRT algorithm for improving random sampling is not covered in this paper.

2.2.1. Rapidly Exploring Random Tree (RRT)

The RRT algorithm is a representative algorithm of the sampling-based path planning method, and was proposed by Steven M. LaValle in 1998 [4]. It is useful for path planning, considering the conditions of non-holonomic constraints, and is designed to have high degrees of freedom [6].

When a random sample is generated in the configuration space, as shown in Figure 4, the node closest to the location of the random sample is found among the nodes constituting the tree with the starting point as the root node. A new node is created at a position away from the node by step length (λ) in the direction of the random sample position and inserted into the tree. If the random sample position is closer than step length, a new node

is created at the random sample position and inserted into the tree. This tree expansion process is repeated until the destination is reached.

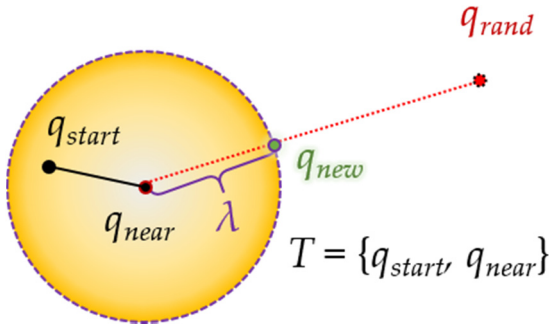


Figure 4. Process of RRT algorithm. A new node is created at location q_{new} separated by step length λ in the direction of the random sampling position q_{rand} based on the random sampling position q_{rand} and the nearest node (position) q_{near} in tree T with start point q_{start} as the root node.

2.2.2. RRT-Connect

As samples appear with the same probability in all regions, path planning through the RRT algorithm may have the disadvantage that the tree easily extends in several random directions irrespective of the destination, resulting in a longer convergence time and inefficiency. The RRT-connect algorithm [23] proposed by James J. Kuffner Jr. in 2000 aims to compensate for this disadvantage by adopting two major new strategies.

The first idea is swapping. It involves designating a starting point and a target point as each root node and alternately extending in the direction of each other. This prevents the tree from extending in a direction independent of its destination and reduces the time required to search for a path.

The second idea is extending. This means that when the tree is extended, it continues to extend to the tree on the other side if no collision with an obstacle occurs. As shown in Figure 5, if there is no collision with an obstacle when a sample is generated, the tree continues to expand as much as the step length in the direction of the opposite tree, so that the destination can be reached faster.

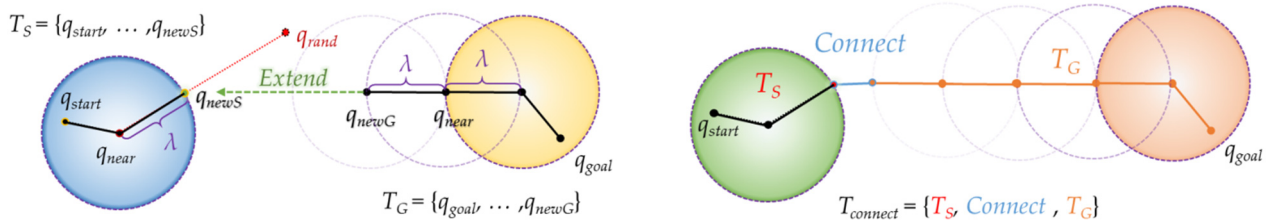


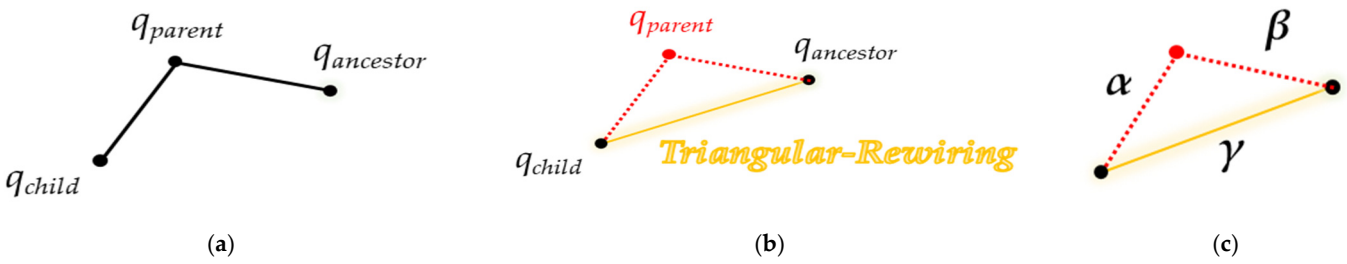
Figure 5. Process of RRT-connect algorithm: (**a**) extend from tree T_b with root as goal position q_{goal} to tree T_a with root as start position q_{start} (T_a's q_{near} extends in T_a's q_{newA} direction); (**b**) as the paths P_a and P_b created in each tree are connected ("Connect") to each other, path P_{merged} is created.

Through this idea, the path planning through the RRT-connect algorithm can find the first complete path at a much higher speed than the existing RRT algorithm.

2.2.3. Triangular-RRT-Connect

Triangular-RRT-Connect [25] is a method that applies the triangular-rewiring method in the extend and connect stages of RRT-connect. Triangular-rewiring uses the principle of triangular inequality. As shown in Figure 6, when there are q_{child}, q_{parent} which is the parent

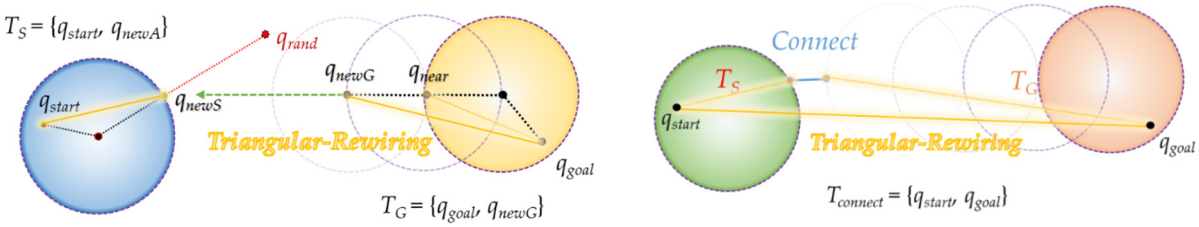
node of q_{child} and $q_{ancestor}$ which is the parent node of q_{parent}, triangular-rewiring eliminates the path through the q_{parent} node when there is no obstacle between q_{child} and $q_{ancestor}$ and connects directly between q_{child} and $q_{ancestor}$.



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

Figure 6. Overview of "Triangular-Rewiring" method: (**a**) example tree, (**b**) result of "Triangular-Rewiring", (**c**) applied trigonometric inequality ($\alpha + \beta > \gamma$).

Figure 7 shows "extend" and "connect" of triangular-RRT-connect. In triangular-RRT-connect, there is a tree extending from the goal point and a tree extending from the start point. In Figure 7a, T_S and T_G are trees extending from the start point and the goal point, respectively. Extend occurs at T_G when sampling is performed at a random location q_{rand} and a new node, q_{newS}, is created based on T_S. At this time, the triangular-rewiring method works on all nodes generated during the "extend" process. Figure 7b is connect in triangular-RRT-connect. After the two trees are connected, the triangular-rewiring method is applied to the merged tree. $T_{connect}$ is a tree where T_S and T_G are connected.



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

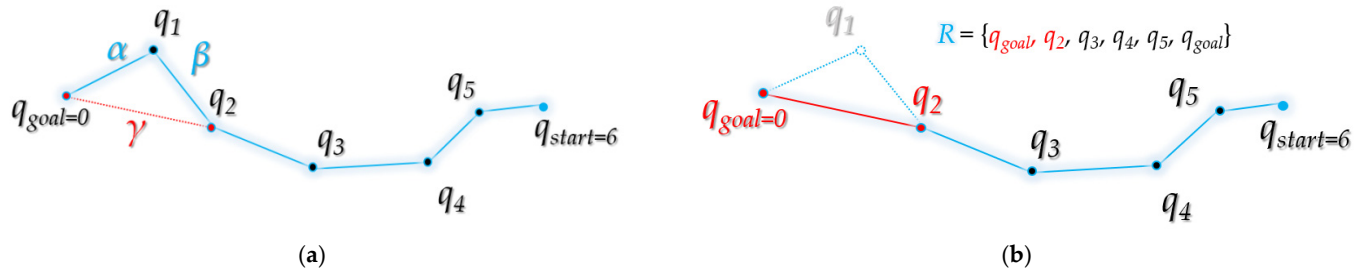
Figure 7. Extend and connect in triangular-RRT-connect: (**a**) extend method in triangular-RRT-connect; (**b**) connect method in triangular-RRT-connect.

3. Bidirectional Interpolation Method for Post-Processing

3.1. Forward Interpolation Process

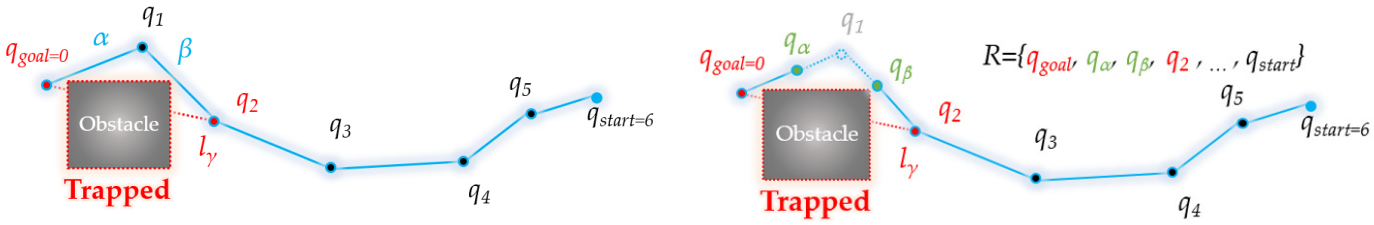
As shown in Figure 8a, when there is no obstacle between the newly inserted node and its grandparent node (collision-free), as shown in Figure 8b, a connection is made between the newly inserted node and its grandparent node, while its parent node is excluded (rewiring). Based on the trigonometric inequality property, this can be corrected to a path that is closer to the optimum than in the existing RRT algorithm.

If the current path is not collision-free from obstacle, as shown in Figure 9, the piecewise linear local path created between the node, its parent node and its grandparent node is optimized through interpolation. In this process, new nodes are interpolated into the existing path to deviate from the piecewise linear path, making it possible to create a smooth path.



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

Figure 8. Overview of rewiring step in forward interpolation process: (**a**) line γ from node q_0 to its grandparent node q_2 in tree R is collision-free (distance: $\gamma < \alpha + \beta$); (**b**) rewiring: grandparent node q_2 of node q_0 is connected to q_0 as parent node, and origin parent node q_1 is excluded from tree R.



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

Figure 9. Overview of interpolation step in forward interpolation process: (**a**) line γ from node q_0 to its grandparent node q_2 in tree R is trapped; (**b**) interpolation: node q_a between node q_0 and node q_1 and node q_b between node q_1 and node q_2 are created. After connecting the parent node of q_0 with q_a, the parent of q_a with q_b and the parent of q_b with q_2, node q_1 is excluded.

Quick-RRT* [10] and triangular-RRT-connect [25] algorithms aim to create a path that is close to optimal using triangular inequality. However, as the node is deleted in the process, the distance between the waypoints on the planned path is longer than that of the RRT algorithm, so the sharp angle on the path line is deepened.

Forward interpolation process is effective for all path planning algorithms, such as RRT algorithm, where optimality is not guaranteed and a local piecewise linear path appears. After the route is planned, it can be applied as a post-processing technique.

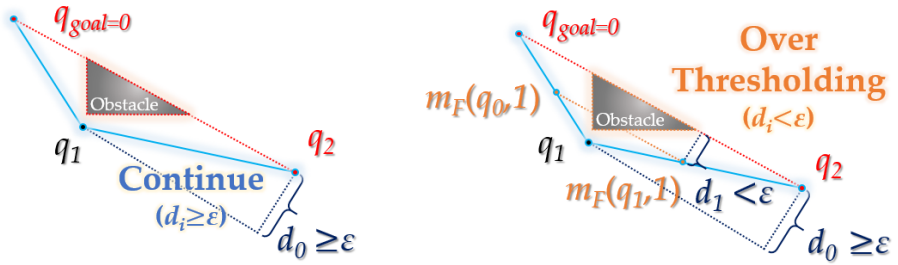
Compared to the classical path planning methods [3], the major advantage of the sampling-based path planning method is the high planning speed owing to a small amount of computation, so there is a prerequisite that the amount of added computation should not be large compared to that required by the existing RRT algorithm.

Forward interpolation process was designed based on the polygon approximation algorithm [18,26]. As shown in Figure 10, the constant value ($\varepsilon > 0$) of ε (the threshold of minimum collision stability) determines how closely the path is approximated to the obstacle.

d_n in Figure 10 follows Equation (1):

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

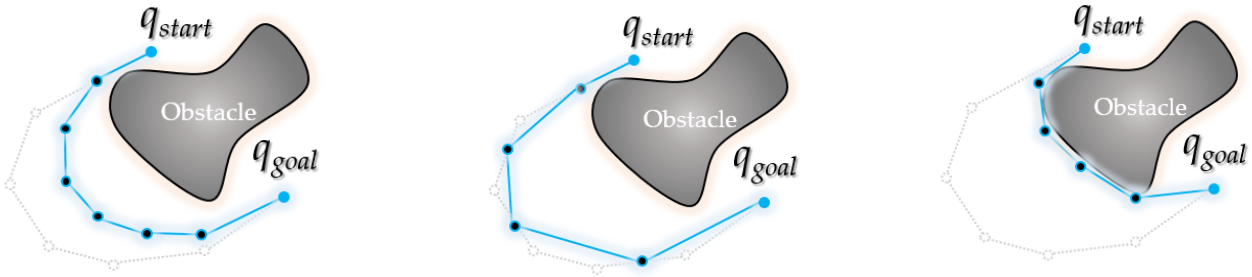
For an arbitrary waypoint q_i, the value of d decreases by $1/2$ as interpolation proceeds (n). The initial value d_0 is the height of the triangle consisting of the length α from q_i to the next waypoint of q_i, the length β from the next waypoint of q_i to the next waypoint of the next waypoint of q_i and the length γ from q_i to the next waypoint of the next waypoint of $q_i (\gamma < \alpha + \beta)$. The value of d_n becomes $(d_n - 1)/2$. As the path gets closer to the obstacle as d gets smaller, it is compared to ε functions as a measure to confirm clearance.



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

Figure 10. Condition of interpolation step at forward interpolation process: (**a**) interpolation continues**:** the height d_0 of a triangle made from waypoint q_0, q_1 and q_2 in random path is higher than ε; (**b**) interpolation break**:** the height d_1 of a triangle made from midpoint $m_F(q_0,1)$ (between q_0 and q_1), node q_1 and midpoint $m_F(q_1,1)$ (between q_1 and q_2) is less than ε.

However, optimality and clearance are conflicting properties; as shown in Figure 11, the smaller the ε value (the minimum value of ε: 0), the higher the optimality and the lower the clearance. Conversely, the larger the ε value, the higher the clearance and the lower the optimality. Therefore, ε should be set to an appropriate value based on the environment.



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

Figure 11. Difference according to ε value in forward interpolation process: (**a**) result when the value of ε is equal to random value n; (**b**) result when ε value is more than random value n; (**c**) result when ε value is less than random value n.

3.1.1. Pseudocode of Forward Interpolation Process

Forward interpolation process is a post-processing method applied after the path is planned in RRT-like algorithms. Mathematical modeling is based on a two-dimensional Euclidean space.

Algorithm 1 presents the pseudocode of forward interpolation process. Two functions can be called internally: post triangular (Algorithm 2) and interpolation (Algorithm 3).

Algorithm 1 Pseudocode of Forward Interpolation Process.

Input:
$R \leftarrow$ path from {*RRT/RRT-Connect/Tri-RRT/Tri-RRT-Connect* / ... }
$C \leftarrow$ 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 *ForwardInterpolationProcess*

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 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** $R \leftarrow postTriangular(R, \varepsilon, t, f_{modify})$
9 **Else**
10 $R \leftarrow interpolation(R, C, \varepsilon, t, f_{modify})$
11 $q_{child} \leftarrow t$-th point in R
12 $q_{parent} \leftarrow$ next point of q_{child} in R
End

Algorithm 2 Pseudocode of the 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** *ForwardInterpolationProcess*
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

Algorithm 3 Pseudocode of the Interpolation Function.

Input:
$R \leftarrow$ path R from *ForwardInterpolationProcess*
$C \leftarrow$ position set C from *ForwardInterpolationProcess*
$\varepsilon \leftarrow$ threshold value ε from *ForwardInterpolationProcess*
$t \leftarrow$ point index t from *ForwardInterpolationProcess*
$f_{modify} \leftarrow$ boolean f_{modify} from *ForwardInterpolationProcess*
Output:
$R \leftarrow$ modified path R
$t \leftarrow$ updated point Index t //**return by reference**
$f_{modify} \leftarrow$ result of boolean f_{modify} //**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** *ForwardInterpolationProcess*
Begin
1 $d \leftarrow$ altitude of 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 ← midpoint between q_parent and q_ancestor
4    While true Do
5        If d >= ε Then
6            If not isTrapped(m_a, m_b, C) Then
7                R ← Delete path<q_child, q_parent> and path<q_parent, q_ancestor> from R
8                R ← Insert path<q_child, m_a>, path<m_a, m_b>, and path<m_b, q_ancestor> to R
9                f_modify ← true
10               Break
11           Else
12               d ← d/2
13               m_a ← midpoint between m_a and q_parent
14               m_b ← midpoint between m_b and q_parent
15       Else
16           t ← t + 1
17           Break
End
```

The input value of forward interpolation process consists of the path R planned through the RRT-like algorithms, the obstacle area information C and the threshold value ε of the minimum clearance.

f_{modify} is a variable that checks 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 path correction does not occur when the process is repeated, the algorithm is terminated. t refers to the index of the waypoint of R that is currently focused. That is, if t is 0, it refers to the starting point, which is the first point of R.

In R, when the first starting point is q_{child}, the next point is q_{parent} and the next point after that point is called $q_{ancestor}$, the algorithm checks whether the line between q_{child} and $q_{ancestor}$ is collision-free (*isTrapped*() function). If it is collision-free, the *postTriangular*() function is called; if not, the *interpolation*() function is called. The *postTriangular*() function connects q_{child} and $q_{ancestor}$, and q_{parent} is excluded from the existing path. The *interpolation*() function finds a random point between (q_{child} and q_{parent}) and (q_{parent} and $q_{ancestor}$) that is collision-free when connected (interpolation), and rewires q_{child}, $q_{ancestor}$ and the two points found. If R and t are updated by the *postTriangular*() or *interpolation*() function, 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 again for the updated q_{child} and $q_{ancestor}$.

Here, the path modification by the *postTriangular*() function deletes the existing waypoints and creates a path that is close to optimal, but has the effect of sharpening the path shape. Path modification by the *interpolation*() function has the effect of creating a path that is close to an optimal path and smoothing the path shape while adding/inserting new waypoints between existing waypoints. For creating a path that is close to optimal, the *postTriangular*() function modifies the path more efficiently than the *interpolation*() function.

3.1.2. Pseudocode of the Post Triangular Function from Forward Interpolation Process

The input value of the *postTriangular*() function consists of the path R, the waypoint index t and f_{modify}, which states whether the path has been modified by forward interpolation process.

Rewiring is performed on the t-th waypoint q_{child} of R, the next point q_{parent} and the next point $q_{ancestor}$ of q_{parent}. First, the path between q_{child} and q_{parent} and the existing path between q_{parent} and $q_{ancestor}$ are deleted. Then, the path between q_{child} and $q_{ancestor}$ is inserted. Finally, f_{modify} returns '*true*' because the path has been modified.

3.1.3. Pseudocode of Interpolation Function from Forward Interpolation Process

As shown in Figure 12, the interpolation of forward interpolation process is performed at three points (random interpolation point (q_0), the next interpolation point (q_1) and the next interpolation point (q_2) of point q_1). It aims to find the interpolation point ($m_F(q_0)$, $m_F(q_1)$) that is collision-free from the obstacle between the waypoints (q_0~q_1, q_1~q_2) while descending in the direction of the midpoint (q_1).

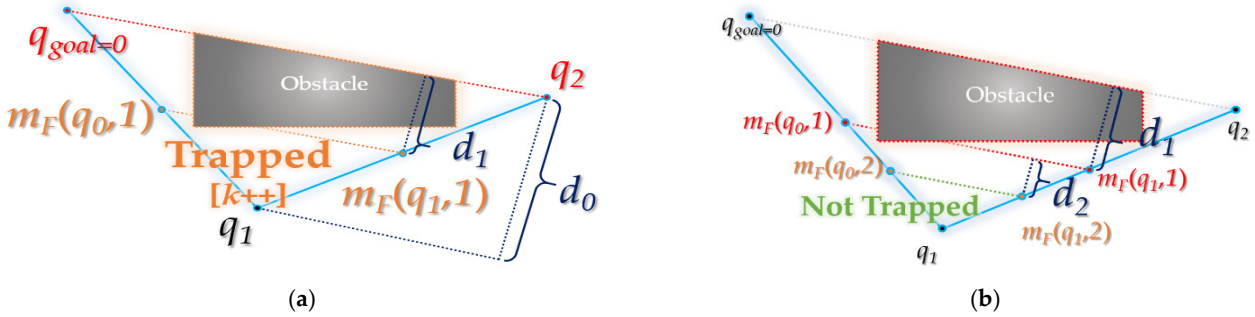


Figure 12. Details of forward interpolation process: (**a**) when the midpoint $m_F(q_0,1)$ of the waypoints q_0, q_1 and the midpoint $m_F(q_1,1)$ of q_1, q_2 are not collision-free from the obstacle; (**b**) when the midpoint $m_F(q_0,2)$ of the midpoint $m_F(q_0,1)$, q_1 and the midpoint $m_F(q_1,2)$ of midpoint $m_F(q_1,1)$, $q1$ are not collision-free from the obstacle.

The interpolation point m_F follows Equations (2) and (3):

$$\xi^n(q_i) := \begin{cases} (\overbrace{\xi \circ \xi \circ \ldots \circ \xi}^{n})(q_i), & n > 0 \\ q_i, & n = 0 \end{cases} \tag{2}$$

First, $\xi()$ is a function that receives a random node as a variable and returns the parent node of that node. The n-th square of $\xi()$ ($n \geq 0$) can be expressed as in Equation (2), when $n = 0$, $\xi^0(q_i) := q_i$ holds.

$$m_F(q_i,k) = \begin{cases} \left(\frac{m_F(q_i,k-1)\cdot x + \zeta(q_i)\cdot x}{2}, \frac{m_F(q_i,k-1)\cdot y + \zeta(q_i)\cdot y}{2}\right), & k > 0 \\ q_i, & k = 0 \end{cases} \quad (k \in \mathbb{N}) \tag{3}$$

When the k-th interpolation point of a random waypoint q_i is $m_F(q_i,k)$, the 0-th interpolation point itself becomes q_i. The first interpolation point is the midpoint of q_i and the next point $\xi(q_i)$ of 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 $\xi(q_i)$. At this time, d becomes $(d_{k-1})/2$.

Algorithm 3 is the pseudocode of the *interpolation*() function in forward interpolation process.

The input value of the *interpolation*() function of forward interpolation process consists of the path R, the obstacle area information C, the waypoint index t and whether to modify the path f_{modify} from in forward interpolation process.

A triangle is made of three waypoints (the t-th waypoint q_{child} of R, the next point q_{parent} of q_{child} and the next point $q_{ancestor}$ of q_{parent}), and the height d of the triangle can be found. 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 collision-free from the obstacle (*isTrapped*()), the existing 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. Furthermore, as the path has been modified, f_{modify} becomes '*true*', returns it, and the function ends. If the distance between m_a and m_b is not collision-free from obstacles, the value of d is 1/2, m_a is the midpoint of m_a and q_{parent}, and m_b is updated to the midpoint of m_b and q_{parent}, so it must be determined whether m_a and m_b are collision-free from obstacles.

This iterative process proceeds until a case is found in which m_a and m_b are collision-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 function is terminated.

3.2. Backward Interpolation Process

As shown in Figure 13, a collision-free interpolation point is found while descending in the direction of the midpoint (q_1) among the three points (a random interpolation point (q_0), the next interpolation point (q_1) of q_0 and the next interpolation point (q_2) of q_1). From the interpolation point, it ascends again in the direction of the obstacle as far as possible ($d >= \varepsilon$), and a waypoint collision-free from obstacle is found between the interpolation point and the waypoint ($m_F(q_0)$~q_1, q_1~$m_F(q_1)$).

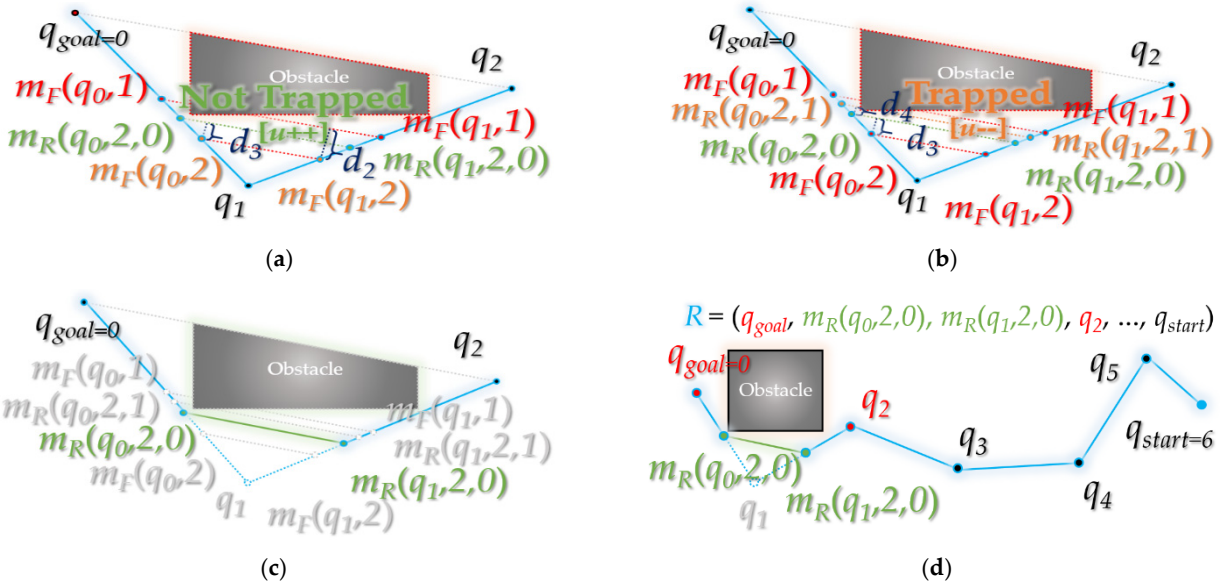


Figure 13. Details of backward interpolation process: (**a**) the path between midpoint $m_R(q_0,2,0)$ of interpolation point $m_F(q_0,1)$, $m_F(q_0,2)$ for existing q_0~q_1 and midpoint $m_R(q_1,2,0)$ of interpolation point $m_F(q_1,2)$, $m_F(q_1,1)$ for existing q_1~q_2 is collision-free; (**b**) the path between midpoint $m_R(q_0,2,1)$ of $m_F(q_0,1)$, $m_R(q_0,2,0)$ and midpoint $m_R(q_1,2,1)$ of $m_R(q_1,2,0)$, $m_F(q_1,1)$ is not collision-free; (**c**) collision-free interpolation point $m_R(q_0,2,0)$, $m_R(q_1,2,0)$ is closest to the obstacle (when $d < \varepsilon$); (**d**) rewiring: interpolation points $m_R(q_0,2,0)$ and $m_R(q_1,2,0)$ are inserted between the existing paths q_0~q_2.

Accordingly, the proposed method can obtain a path that is close to optimal compared to the existing PTPMI [17] method.

The interpolation point m_r follows Equation (4):

$$m_R(q_i,k,u) = \begin{cases} \left(\frac{m_R(q_i,k,u-1)\cdot x + m_F(q_i,k-1)\cdot x}{2}, \frac{m_R(q_i,k,u-1)\cdot y + m_F(q_i,k-1)\cdot y}{2}\right), & u > 0 \\ \left(\frac{m_F(q_i,k)\cdot x + m_F(q_i,k-1)\cdot x}{2}, \frac{m_F(q_i,k)\cdot y + m_F(q_i,k-1)\cdot y}{2}\right), & u = 0 \end{cases} \quad (u \in \mathbb{N}) \tag{4}$$

The u-th interpolated point in the direction of the obstacle from the k-th interpolation point $m_F(q_i,k)$ of a random waypoint q_i is called $m_R(q_i, k, u)$ (the value of k is fixed). At this time, if u is 0, it becomes the midpoint of $m_F(q_i, k)$ and $m_F(q_i, k-1)$. If u is 1, it is the midpoint of $m_R(q_i, k, 0)$ and $m_F(q_i, k-1)$. That is, $m_R(q_i, k, u)$ ($u > 0$) becomes the midpoint between $m_R(q_i, k, u-1)$ and $m_F(qi, k-1)$. For reference, d becomes $(d_{k+u-1})/2$. Here, (when u is 0) $m_F(q_i, k)$ (and $m_F(\xi^2(q_i),k)$) goes down in the $\xi(q_i)$ direction and becomes the first obstacle collision point. $m_F(q_i, k-1)$ (and $m_F(\xi^2(q_i), k-1)$) is the point at which it did not collide with the last obstacle. Therefore, Equation (4) interpolates within the region between the obstacle collision point and the obstacle non-impact collision point.

$m_R(q_i, k, u)$ in Equation (4) can also be expressed as Equation (7) through Equations (5) and (6):

$$\frac{m_F(q_i,k) + m_F(q_i,k-1)}{2} = \frac{3(m_F(q_i,k)) - \zeta(q_i)}{2} \tag{5}$$

$$\frac{m_R(q_i,k,u-1) + m_F(q_i,k-1)}{2} = \frac{3(m_R(q_i,k,u-1)) - m_R(q_i,k,u-2)}{2} \tag{6}$$

It starts with $m_F(q_i,k)$ and $\xi(q_i)$ when u = 0. Then, $m_R(q_i,k,u)$ is found as a point that divides the line segment connecting the previous two points $m_R(q_i,k,u-1)$ and $m_R(qi,k,u-2)$ in a 3:1 ratio (k value is fixed).

$$\therefore m_R(q_i,k,u) = \begin{cases} \left(\frac{3(m_R(q_i,k,u-1)\cdot x) - m_R(q_i,k,u-2)\cdot x}{2}, \frac{3(m_R(q_i,k,u-1)\cdot y) - m_R(q_i,k,u-2)\cdot y}{2}\right), & u > 0 \\ \left(\frac{3(m_F(q_i,k)\cdot x) - \zeta(q_i)\cdot x}{2}, \frac{3(m_F(q_i,k)\cdot y) - \zeta(q_i)\cdot y}{2}\right), & u = 0 \end{cases} \tag{7}$$

In the end, Equation (7) shows the same result as Equation (4), and it is more efficient in terms of the space complexity of the algorithm.

Pseudocode Backward Interpolation Process

Algorithm 4 presents the pseudocode of the *interpolation*() function of backward interpolation process.

Algorithm 4 Pseudocode of Backward Interpolation Process.

Input:
$R \leftarrow$ path R from *ForwardInterpolationProcess*
$C \leftarrow$ position set C from *ForwardInterpolationProcess*
$\varepsilon \leftarrow$ threshold value ε from *ForwardInterpolationProcess*
$t \leftarrow$ point index t from *ForwardInterpolationProcess*
$f_{modify} \leftarrow$ boolean f_{modify} from *ForwardInterpolationProcess*
Output:
$R \leftarrow$ modified path R
$t \leftarrow$ updated point index t **//return by reference**
$f_{modify} \leftarrow$ result of boolean f_{modify} **//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** *ForwardInterpolationProcess*
Begin
$d \leftarrow$ altitude of triangle consisting of q_{child}, q_{parent}, and $q_{ancestor}$ with base<q_{child}, $q_{ancestor}$>
$m_a \leftarrow$ midpoint between q_{child} and q_{parent}
$m_b \leftarrow$ midpoint between q_{parent} and $q_{ancestor}$
While ***true*** **Do**
    If d **>=** ε **Then**
        If not *isTrapped*(m_a, m_b, C) **Then**
            $R \leftarrow$ **Delete** path<q_{child}, q_{parent}> and path<q_{parent}, $q_{ancestor}$> from R
            $m_{backA} \leftarrow$ external division point of line segment<q_{parent}, m_a> with the ratio 3:1
            $m_{backB} \leftarrow$ external division point of line segment<q_{parent}, m_b> with the ratio 3:1
            While ***true*** **Do**
                $m_{freeA} \leftarrow m_a$
                $m_{freeB} \leftarrow m_b$
                If not *isTrapped*(m_{backA}, m_{backB}, C) **Then**
                    $m_a \leftarrow m_{backA}$

```
15                                          m_b ← m_backB
16                                  Else    Break
17                                  d ← d/2
18                                  If not d >= ε Then Break
                                    m_backA ← external division point of line segment <m_freeA,
19  m_a> with
    the ratio 3:1
                                    m_backB ← external division point of line segment <m_freeB,
20  m_b> with
    the ratio 3:1
21                          R ← Insert path<q_child, m_a>, path<m_a, m_b> and path<m_b, q_ancestor> to R
22                          f_modify ← true
23                          Break
24                  Else
25                          d ← d/2
26                          m_a ← midpoint between m_a and q_parent
27                          m_b ← midpoint between m_b and q_ancestor
28              Else
29                  t ← t + 1
30                  Break
End
```

The input value of the *interpolation*() function of backward interpolation process consists of path R, obstacle area information C, waypoint index t and path modification f_{modify}.

Compared to the *interpolation*() function in forward interpolation process, lines 8–20 have been inserted in this *interpolation*() function. These contents are interpolated again in the direction of the obstacle after the unidirectional (q_{parent} direction) interpolation is completed (when collision-free from the obstacle). From the 8th line, m_{backA} is the point where the line segment connecting q_{parent} and m_a is externalized in a 3:1 ratio, m_{backB} is the point where the line connecting q_{parent} and m_b is externalized in a 3:1 ratio, m_{freeA} is m_a and m_{freeB} is m_b. If the route between m_{backA} and m_{backB} is collision-free from the obstacle (*isTrapped*()), m_a is updated to m_{backA} and m_b to m_{backB}. If it is not collision-free from the obstacle, based on the current m_a and m_b, a path connecting q_{child} and m_a, a path connecting m_a and m_b and a path connecting m_b and $q_{ancestor}$ are inserted, and the function is terminated. In the case of being collision-free from the obstacle, if m_a and m_b are updated, d becomes $1/2$. If d is less than ε, the value of t is incremented by 1 and the function terminates. Otherwise, m_{backA} is updated to the point where the line segment connecting m_{freeA} and m_a is externalized in a 3:1 ratio, m_{backB} is updated to the point where the line segment connecting m_{freeB} and m_b is externalized in a 3:1 ratio, m_{freeA} is updated to m_a and m_{freeB} to m_b, and the previous process is repeated.

3.3. Overview of Bidirectional Interpolation Method

Figure 14 shows the overall flowchart of bidirectional interpolation method. Here, $\xi^{t}(q_{goal})$ means the t-th next waypoint from the starting point q_{goal} of the path R, and $\xi^{t+n}(q_{goal})$ means the n-th next waypoint in the $\xi^{t}(q_{goal})$. That is, there are n waypoints between $\xi^{t}(q_{goal})$ and $\xi^{t+n}(q_{goal})$.

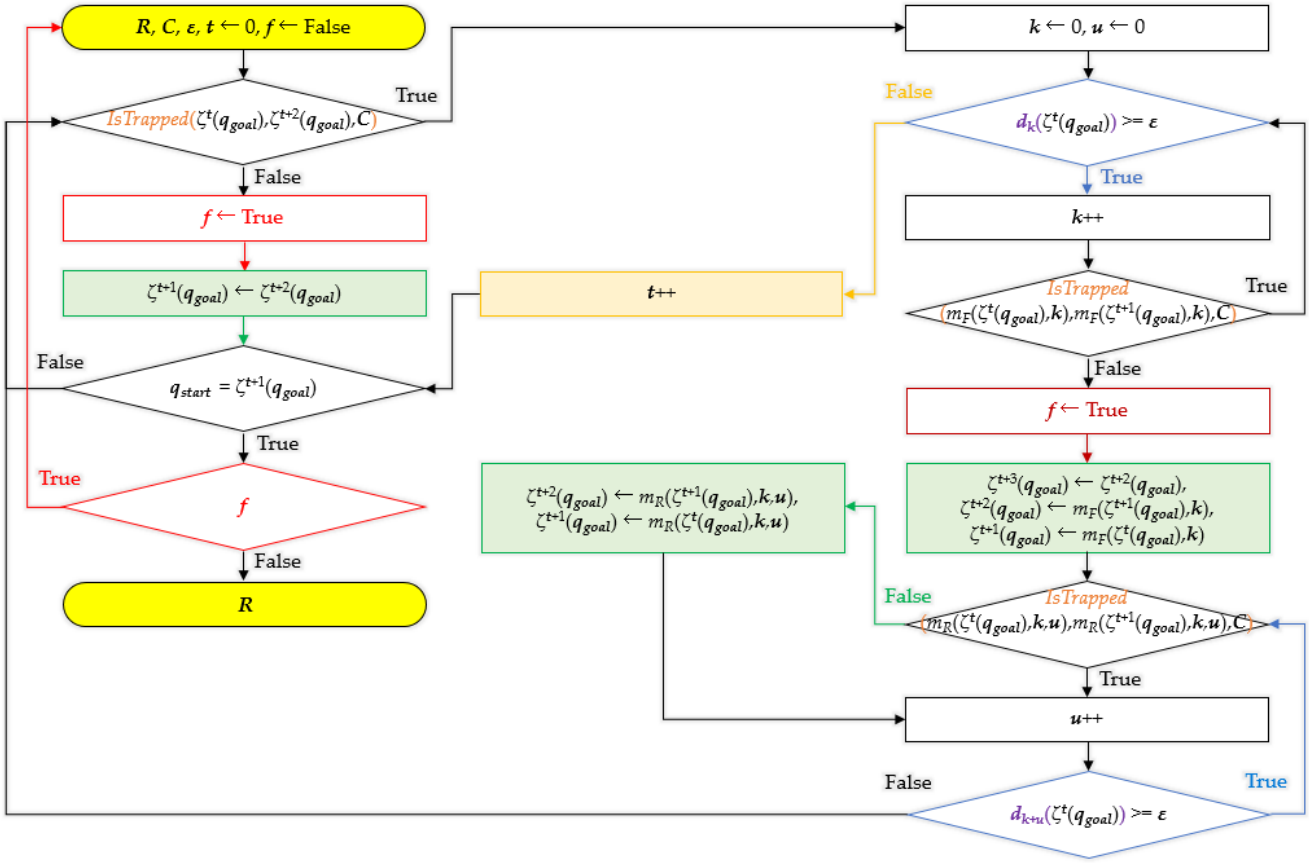


Figure 14. Flowchart of bidirectional interpolation method.

Figure 15 shows, in detail, the intermediate steps in the process followed by bidirectional interpolation method to correct the planned path R from the starting point q_{goal} to the destination point q_{start}.

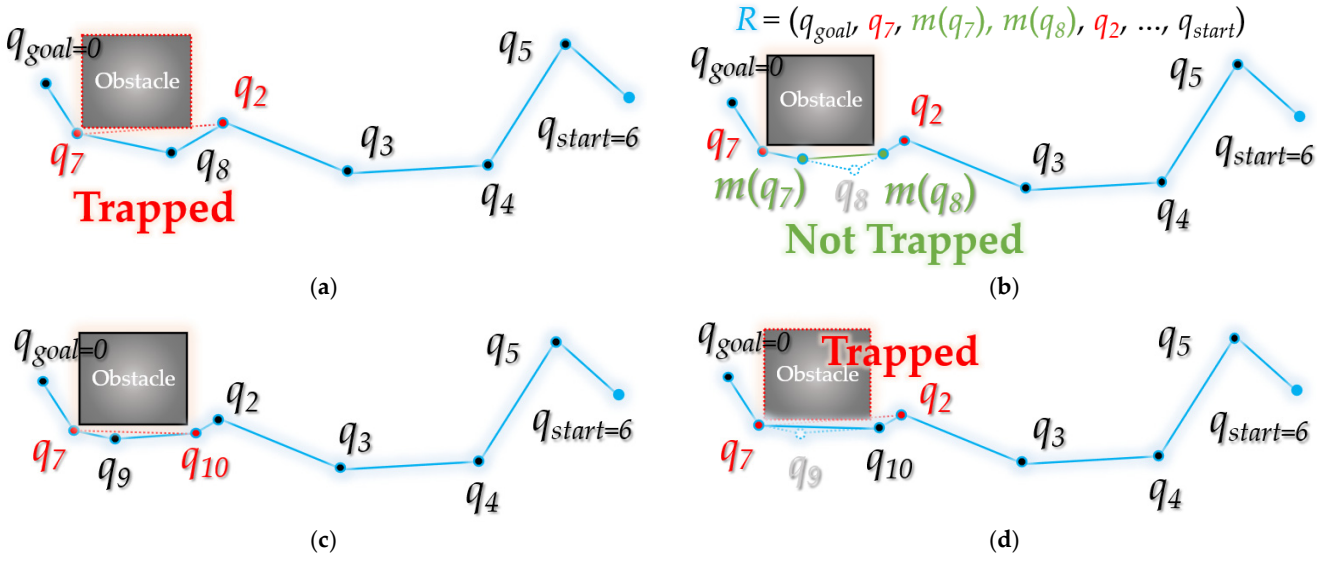


Figure 15. *Cont.*

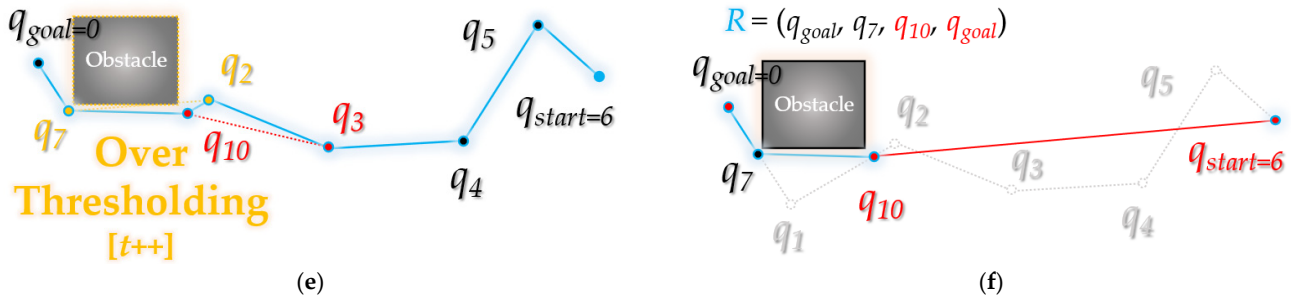


Figure 15. Process of bidirectional interpolation method: (**a**) (t = 1) between the waypoints q_7 and q_2 is not collision-free from the obstacle; (**b**) find the interpolation point $m(q_7)$ between q_7 and q_8 and the interpolation point $m(q_8)$ between q_8 and q_2, insert the interpolation points between the paths and delete the q_8 path; (**c**) the route between q_7 and q_{10} (from $m(q_8)$) is collision-free from the obstacle, so join the path and delete path q_9 (from $m(q_7)$); (**d**) the route between q_7 and q_2 is not collision-free from the obstacle; (**e**) in the process of interpolation between q_7 and q_2, assuming that d becomes smaller than ε, move the focusing point ($\xi^t(q_{goal})$) to the next point ($t \leftarrow t + 1$) and (t = 2). The route between q_{10} and q_3 is collision-free from the obstacle; (**f**) as it is collision-free from the obstacle from q_{10} to q_{start}, q_{10} and q_{start} are connected, the waypoint between them is deleted.

Figure 15a starts when the waypoint index t is 1. That is, as $\xi^t(q_{goal})$ is $\xi(q_{goal})$, it becomes q_7 in the figure. As q_7 and the next waypoint q_2 are not collision-free from the obstacle, interpolation proceeds. Figure 15b shows that the interpolation point $m(q_7)$ of q_7~q_8 and the interpolation point $m(q_8)$ of q_8~q_2 are free from obstacle collision. This is a case where the vertical distance d between the obstacle and the line segment formed by the interpolation points is smaller than the set threshold ε. The interpolation points $m(q_7)$ and $m(q_8)$ are inserted between the existing paths q_7 to q_2, and the path is modified. Existing paths q_7 to q_8 and q_8 to q_2 are deleted, and paths q_7 to $m(q_7)$, $m(q_7)$ to $m(q_8)$, and $m(q_8)$ to q_2 are inserted. In Figure 15c, the existing paths q_7~q_9, q_9~q_{10} are deleted and q_7~q_{10} is inserted because the distance between q_7 and q_{10} is free from obstacle collision. In this case, q_9 and q_{10} refer to $m(q_7)$ and $m(q_8)$ in Figure 15b. In Figure 15d, interpolation is performed for q_7~q_{10} and q_{10}~q_2 because the line between q_7 and q_2 is not collision-free from the obstacle. Accordingly, index t becomes 2, and the focused waypoint becomes q_{10}, which is $\xi^2(q_{goal})$. At this time, it can be seen that the space between q_{10} and q_3 is free from obstacle collision. Figure 15f shows that all the waypoints on the path q_{10}~q_{start} are free from obstacle collision, so the existing path between q_{10}~q_{start} is deleted, and a path that connects q_{start} in a straight line is inserted in q_{10}. Finally, the path R is modified to (q_{goal}, q_7, q_{10}, q_{goal}).

4. Experimental Results

To check the performance of the bidirectional interpolation method proposed in this paper, the path planning results between visibility graph, RRT-connect, PTPMI and bidirectional interpolation method upon various environments were compared through simulation.

The performance measure is the average path length (px) and planning time (ms) until the first complete path is created when each algorithm (excluding visibility graph algorithm) is repeated 100 times (sampling location changes with every trial).

4.1. Experimental Environment

This section introduces the environment map used in the simulation and the computer specifications (i.e., hardware) for the simulation.

Figure 16 shows the six environmental maps used in the experiment. Here, the green circle (S) refers to the starting point, and the purple circle (G) refers to the destination point. A black polygon with a yellow border (blue in the experimental results) indicates an obstacle. The size of all environment maps is 600 × 600 px, and the step length is 30 px.

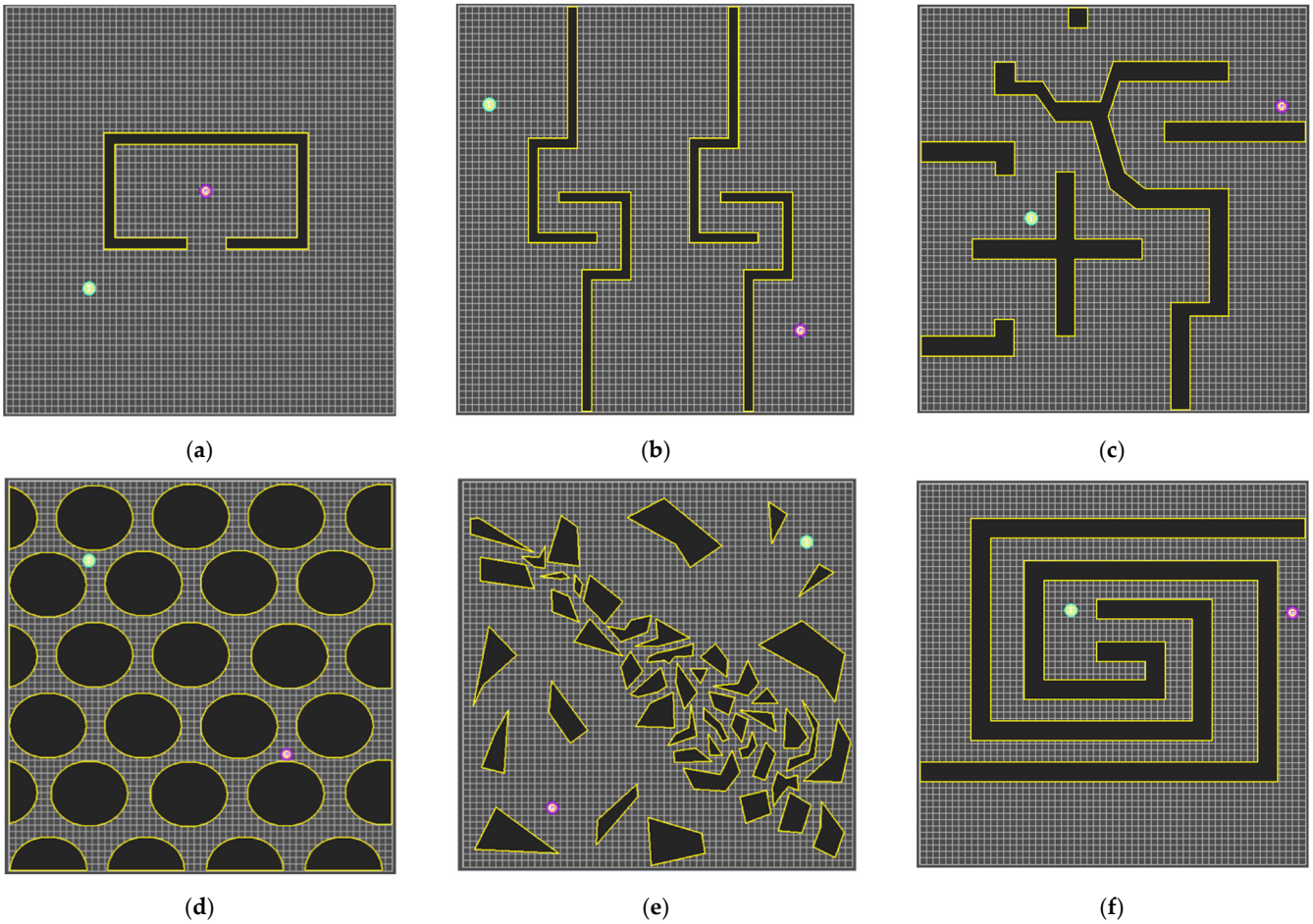


Figure 16. Environmental maps for the experiment: (**a**) Map 1; (**b**) Map 2; (**c**) Map 3; (**d**) Map 4; (**e**) Map 5; (**f**) Map 6.

Various environmental maps were considered and utilized to gauge the performance of bidirectional interpolation method. The environment maps used are important because the results of the performance measurement expected during the experiment are different depending on their composition, i.e., the number, arrangement and shape of the obstacles within the map. In this paper, the six environmental maps shown in Figure 16 are used to verify the performance of bidirectional interpolation method. These maps are part of the experimental environment [27] proposed by Jihee Han in 2017, and the following characteristics and efficiency of performance measures are expected for each map. Figure 16a shows Map 1, an environment in which the completeness of the path planning method can be easily verified, which is also an environment mainly used to show the local minima problem solving in the potential field algorithm [28]. Figure 16b shows Map 2, in which the optimality and completeness of the path planning method can be verified. Figure 16c shows Map 3, which is suited to verifying the optimality and completeness. Figure 16d shows Map 4, which is suited to verifying the optimality of the path planning method, as well as the planning time, because it is composed of obstacles (50 squares) that resemble a curved shape. Figure 16e shows Map 5, which is an environment in which it is easy to comprehensively verify the optimality and completeness of the path planning method as well as the planning time. Figure 16f shows Map 6, in which it is easy to verify the completeness and planning time of the path planning method. Furthermore, Map 6 is an environment that is unfavorable to sampling-based path planning methods such as the RRT algorithm.

As the sampling-based path planning method relies on probabilistic completeness, the number of sampling times and the planning time required increase considerably as there are narrow or few entrances in the direction to the destination.

Table 1 summarizes the performance of the computer used in the simulation. The simulator used for the simulation 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. There may be differences in planning time during simulation depending on computer performance. Therefore, in the experiment in this study, the planning time is compared not absolutely but relatively, based on the RRT-connect algorithm.

Table 1. Computer specifications for the simulation.

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

4.2. Experimental Results and Analysis for Each Map

In this section, the experimental results of applying the algorithms to the environment map presented in Figure 16 are stated and analyzed. The algorithms used are visibility graph, RRT-connect, triangular-RRT-connect, PTPMI and the proposed algorithm. PTPMI and the proposed algorithm were applied to the path created by RRT-connect, and the ε value was set to 50, 30 and 10 px, respectively. As ε requires a higher amount of computation as it decreases, it was set to an appropriate value nearby depending on the step length (30 px) of the experimental environment.

The experimental results will show the planned path for each algorithm for each map and show the piecewise linear shape of the path. In addition, as the results of the visibility graph for each map are presented together, the optimality of each algorithm could be visually confirmed.

The contents to be checked through the table are the path length and planning time, which are performance measures. The length of the path created through each algorithm and the relative ratio for the length of the path created by the visibility graph were considered.

PTPMI and the proposed algorithm are methods for post-processing the generated path. In this study, given that PTPMI and the proposed algorithm are applied based on RRT-connect, the planning time is compared based on the RRT-connect algorithm. The planning time is compared to basic RRT-connect and the difference between planning times of PTPMI and the proposed algorithm must be checked. The path length and planning time are presented in Table 2.

Table 2. Experimental result based on RRT-connect of Map 1 (the parentheses to the right of each value of path length are the relative ratios based on visibility graph (253 px)).

Performance	**RRT-Connect**	**PTPMI Method**			**Bidirectional Interpolation Method**		
		ε: 50 px	**ε: 30 px**	**ε: 10 px**	**ε: 50 px**	**ε: 30 px**	**ε: 10 px**
Path length (px)	379 (150%)	283 (112%)	264 (104%)	258 (102%)	278 (110%)	263 (104%)	257 (101%)
Planning time (ms)	<0	<0	<0	<0	1	<0	<0

Figure 17 shows the path planning results for Map 1 for each algorithm. Looking at the generated path (yellow line), compared to Figure 17a, which is the result of RRT-connect, when PTPMI (Figure 17d–f) and the proposed algorithm (Figure 17g–i) were each applied for post-processing, the piecewise linear shape with sharp curves was reduced. In addition, it can be seen that the smaller the ε value, the higher the similarity with the path generated by the visibility graph (Figure 15c).

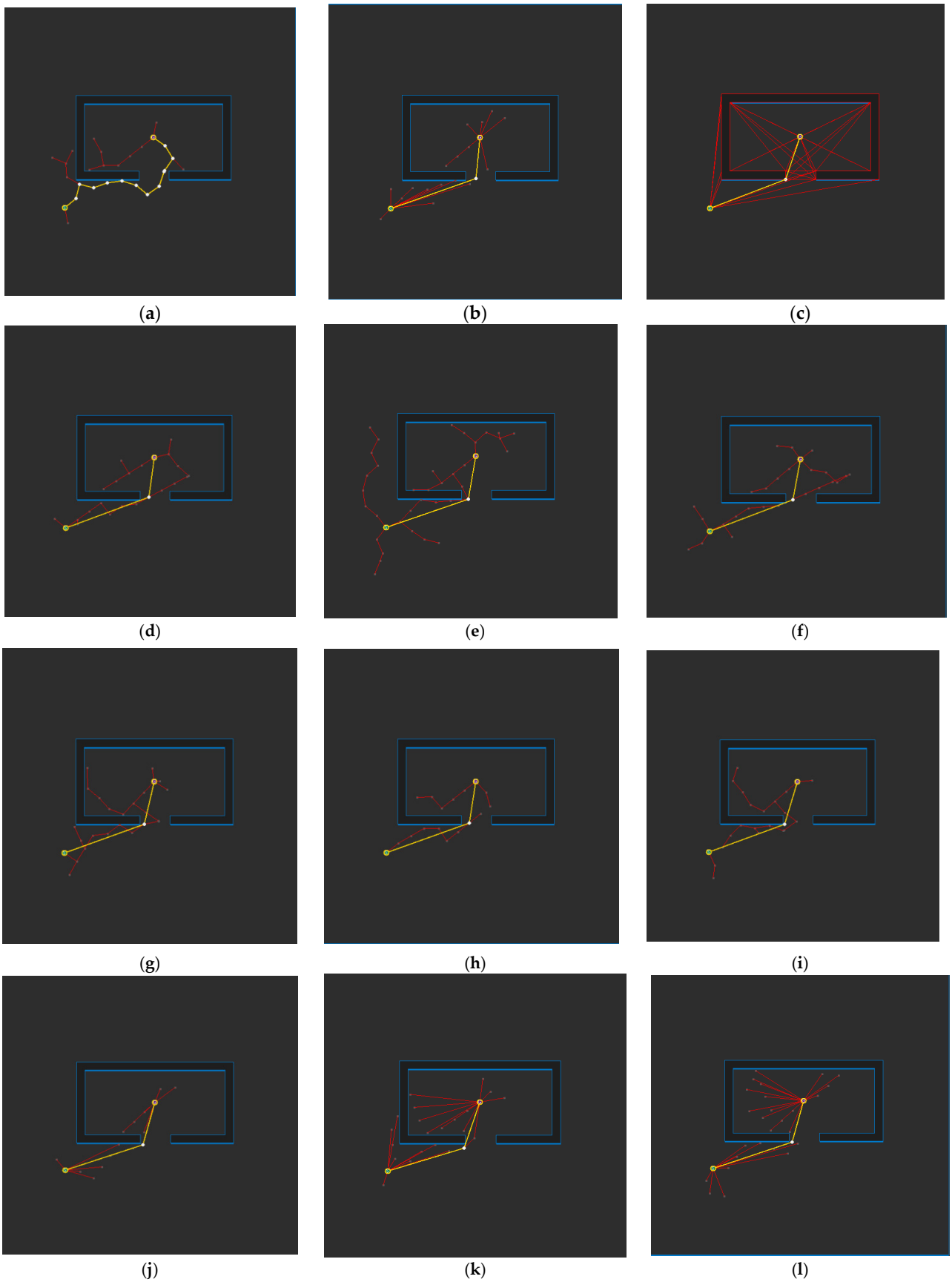
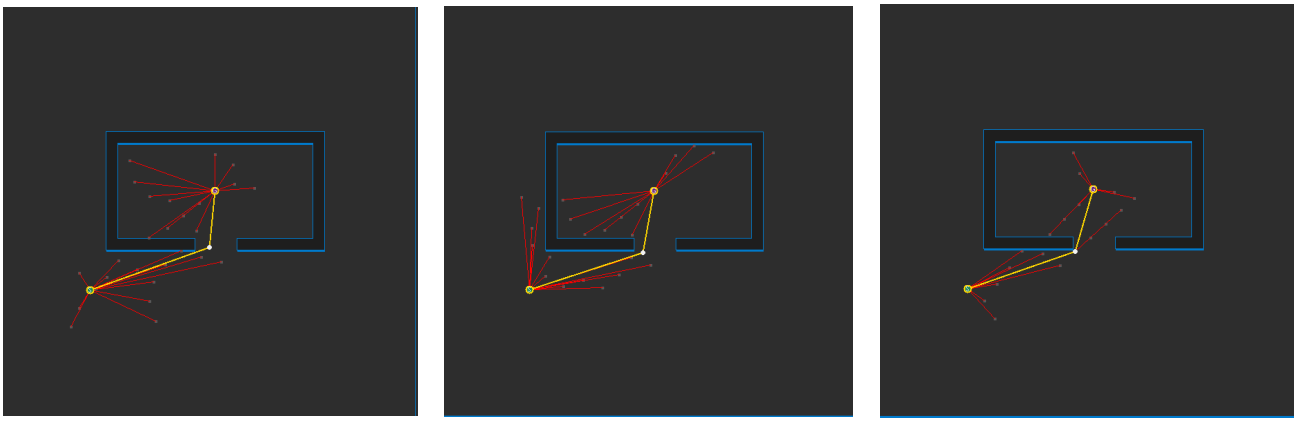


Figure 17. *Cont.*



(**m**) (**n**) (**o**)

Figure 17. Experimental result of Map 1: (**a**) RRT-connect; (**b**) triangular-RRT-connect (**c**) visibility graph; (**d**) RRT-connect PTPMI (ε: 50 px); (**e**) RRT-connect PTPMI (ε: 30 px); (**f**) RRT-connect PTPMI (ε: 10 px); (**g**) RRT-connect bidirectional interpolation method (ε: 50 px); (**h**) RRT-connect bidirectional interpolation method (ε: 30 px); (**i**) RRT-connect bidirectional interpolation method (ε: 10 px); (**j**) triangular-RRT-connect PTPMI (ε: 50 px); (**k**) triangular-RRT-connect PTPMI (ε: 30 px); (**l**) triangular-RRT-connect PTPMI (ε: 10 px); (**m**) triangular-RRT-connect bidirectional interpolation method (ε: 50 px); (**n**) triangular-RRT-connect bidirectional interpolation method (ε: 30 px); (**o**) triangular-RRT-connect bidirectional interpolation method (ε: 10 px).

Table 2 summarizes the experimental results numerically for Map 1 based on RRT-connect among the presented environmental maps. It can be seen that the path length is the shortest when the ε value of the proposed algorithm is 10 (px) and is also closest to the path generated by the visibility graph (relative ratio is 257 (px)/253 (px), which is about 101%). The planning time was approximately 1 ms when the ε value of the proposed algorithm was 50 (px), and it takes less than 1 ms in most cases except for this case, similar to the standard RRT-connect. Thus, in Map 1, it can be confirmed that the proposed algorithm is more efficient and optimal than PTPMI.

Table 3 summarizes the experimental results numerically for Map 1 based on triangular-RRT-connect. It can be seen that the path length is the shortest when the ε value of the proposed algorithm is 10 (px) and is also closest to the path generated by the visibility graph (relative ratio is 257 (px)/253 (px), which is about 101%). The planning time takes less than 1 ms in all cases, similar to the standard RRT-connect. Thus, in Map 1, it can be confirmed that the proposed algorithm is a little more efficient and optimal than PTPMI.

Table 3. Experimental result based on triangular-RRT-connect of Map 1 (the parentheses to the right of each value of path length are the relative ratios based on visibility graph (253 px)).

Performance	Triangular-RRT-Connect	PTPMI Method			Bidirectional Interpolation Method		
		ε: 50 px	ε: 30 px	ε: 10 px	ε: 50 px	ε: 30 px	ε: 10 px
Path length (px)	282 (111%)	277 (109%)	264 (104%)	258 (102%)	274 (108%)	264 (104%)	257 (101%)
Planning time (ms)	<0	<0	<0	<0	<0	<0	<0

Figure 18 shows the path planning results for Map 2 for each algorithm. Looking at the generated path (yellow line), compared to Figure 18a, which is the result of RRT-connect, when PTPMI (Figure 18d–f) and the proposed algorithm (Figure 18g–i) were each applied for post-processing, the piecewise linear shape with sharp curves was reduced. In addition, it can be seen that the smaller the ε value, the higher the similarity with the path generated by the visibility graph (Figure 16c).

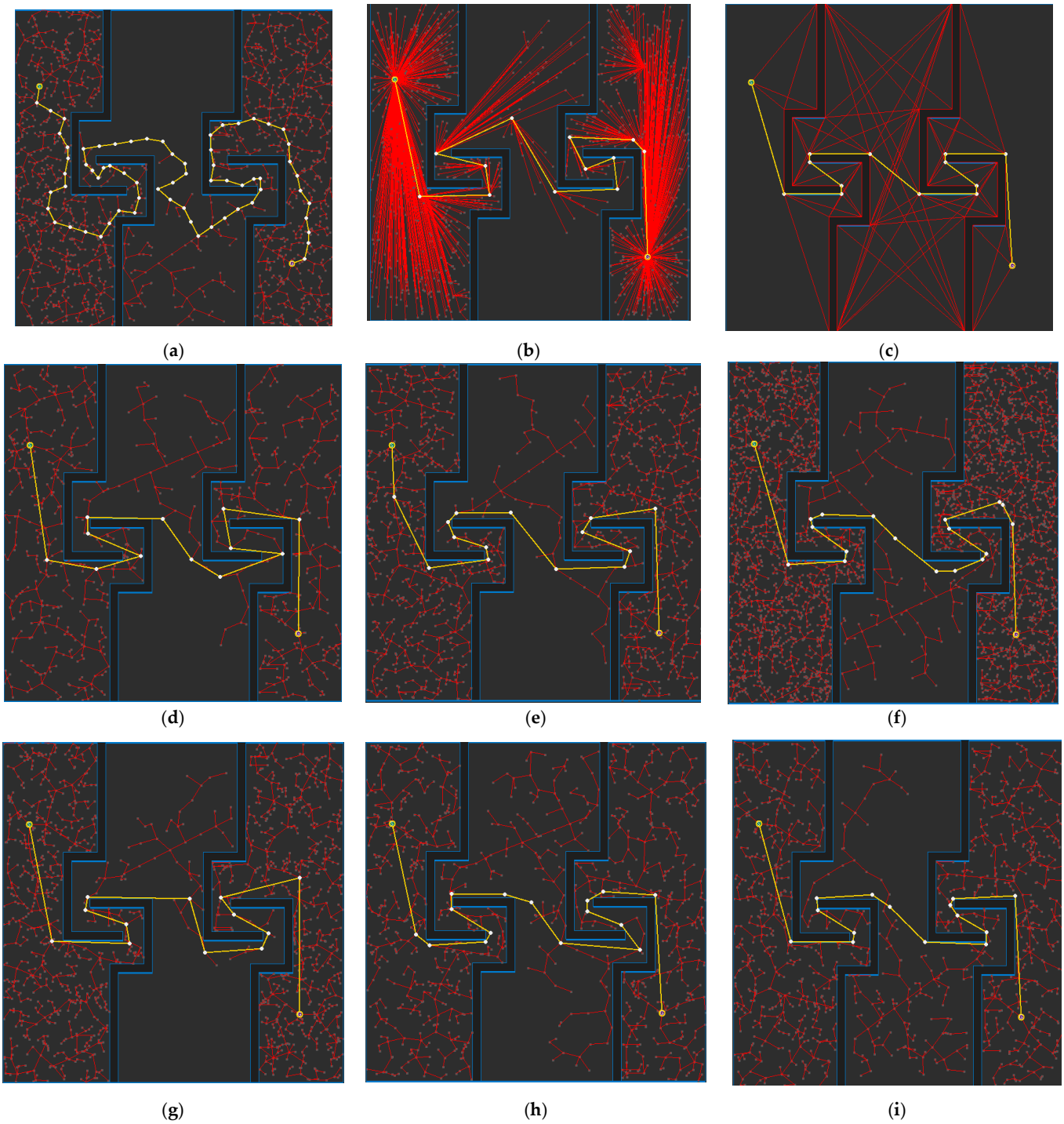


Figure 18. *Cont.*

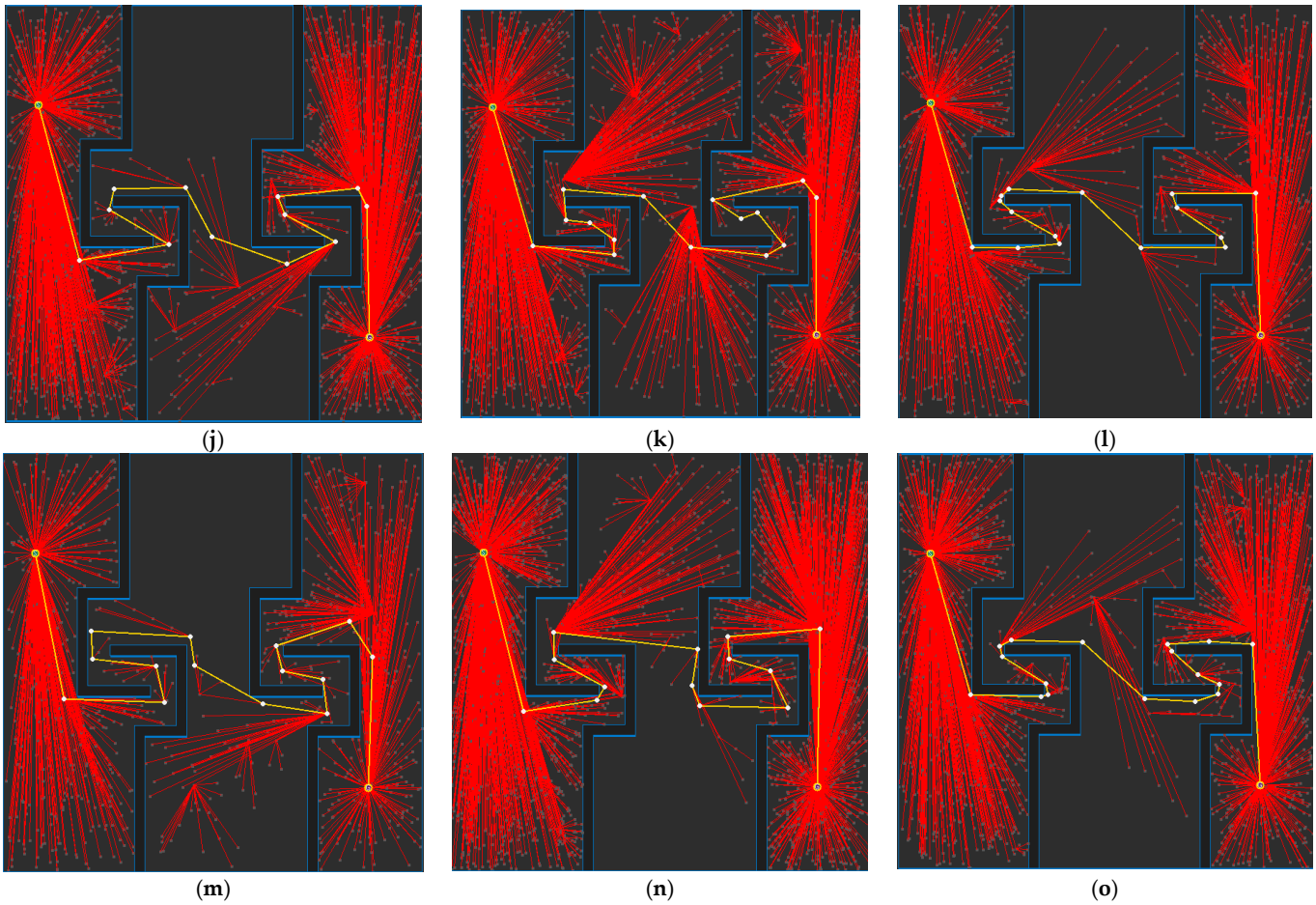


Figure 18. Experimental result of Map 2: (**a**) RRT-connect; (**b**) triangular-RRT-connect (**c**) visibility graph; (**d**) RRT-connect PTPMI (ε: 50 px); (**e**) RRT-connect PTPMI (ε: 30 px); (**f**) RRT-connect PTPMI (ε: 10 px); (**g**) RRT-connect bidirectional interpolation method (ε: 50 px); (**h**) RRT-connect bidirectional interpolation method (ε: 30 px); (**i**) RRT-connect bidirectional interpolation method (ε: 10 px); (**j**) triangular-RRT-connect PTPMI (ε: 50 px); (**k**) triangular-RRT-connect PTPMI (ε: 30 px); (**l**) triangular-RRT-connect PTPMI (ε: 10 px); (**m**) triangular-RRT-connect bidirectional interpolation method (ε: 50 px); (**n**) triangular-RRT-connect bidirectional interpolation method (ε: 30 px); (**o**) triangular-RRT-connect bidirectional interpolation method (ε: 10 px).

Table 4 summarizes the experimental results numerically for Map 2 based on RRT-connect. The path length is the shortest when the ε value of the proposed algorithm is 10 (px) and is closest to the visibility graph (relative ratio is 1223/1172, which is about 104%). It can be seen that the planning time takes longer than RRT-connect when post-processing techniques are applied. However, as expected, as the value of ε decreased, PTPMI and the proposed algorithm did not take longer. Rather, it can be seen that in the proposed algorithm, which requires more steps than PTPMI, the case where ε: 10 (px) (which is expected to take the longest time) shows the smallest difference from RRT-connect. This deviation in planning time is not due to any issues related to the post-processing technique, but is due to the random sampling effect of the RRT-like algorithms. In other words, it is difficult to find a solution for Map 2 using the RRT-based algorithm. In summary, in Map 2, it was confirmed that the proposed algorithm guarantees the optimality of the path length compared to other algorithms.

Table 4. Experimental result based on RRT-connect of Map 2 (the parentheses to the right of each value of path length are relative ratios based on visibility graph (1172 px)).

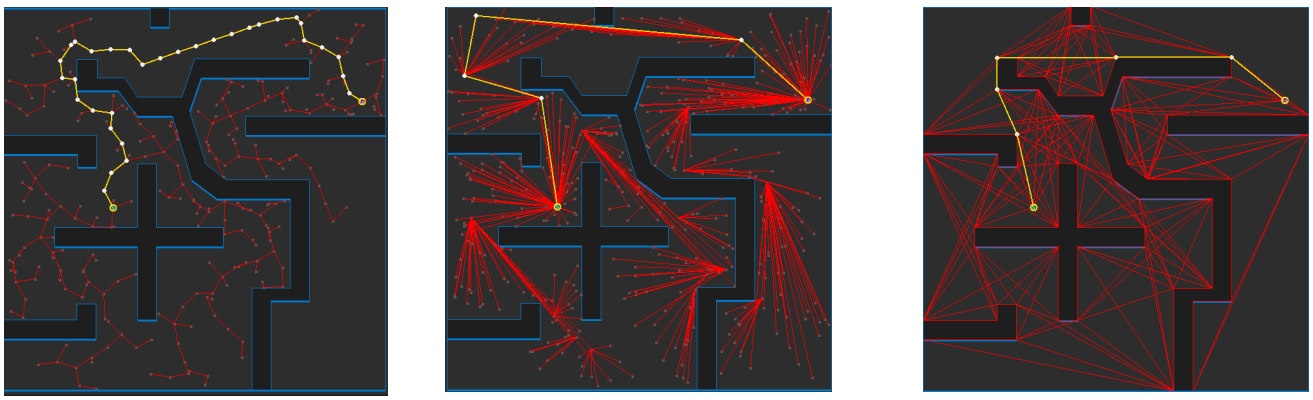
Performance	RRT-Connect	PTPMI Method			Bidirectional Interpolation Method		
		ε: 50 px	ε: 30 px	ε: 10 px	ε: 50 px	ε: 30 px	ε: 10 px
Path length (px)	1843 (157%)	1399 (119%)	1326 (113%)	1230 (105%)	1395 (119%)	1324 (112%)	1223 (104%)
Planning time (ms)	220	242	264	250	243	272	223

Table 5 summarizes the experimental results numerically for Map 2 based on triangular-RRT-connect. The path length is the shortest when the ε value of the proposed algorithm is 10 (px) and is closest to the visibility graph (relative ratio is 1229/1172, which is about 105%). However, as expected, as the value of ε decreased, PTPMI and the proposed algorithm did not take longer. Even some results (ε: 30 px, 10 px) have a shorter planning time than triangular-RRT-connect. Rather, it can be seen that in the proposed algorithm, which requires more steps than PTPMI, the case where ε: 10 (px) (which is expected to take the longest time) shows shorter time than triangular-RRT-connect. This difference in planning time is not due to any issues related to the post-processing technique, but is due to the random sampling effects of the RRT-like algorithms. In other words, it is more difficult to find a solution for Map 2 using the RRT-based algorithm. In summary, in Map 2, it was confirmed that the proposed algorithm guarantees the optimality of the path length compared with other algorithms.

Table 5. Experimental result based on triangular-RRT-connect of Map 2 (the parentheses to the right of each value of path length are relative ratios based on visibility graph (1172 px)).

Performance	Triangular-RRT-Connect	PTPMI Method			Bidirectional Interpolation Method		
		ε: 50 px	ε: 30 px	ε: 10 px	ε: 50 px	ε: 30 px	ε: 10 px
Path length (px)	1478 (126%)	1405 (120%)	1331 (113%)	1230 (105%)	1404 (120%)	1331 (113%)	1229 (105%)
Planning time (ms)	195	197	194	214	205	181	194

Figure 19 shows the path planning results for Map 3 for each algorithm. Looking at the generated path (yellow line), compared to Figure 19a, which is the result of RRT-connect, when PTPMI (Figure 19d–f) and the proposed algorithm (Figure 19g–i) were each applied for post-processing, the piecewise linear shape with sharp curves was reduced. In addition, it can be seen that the smaller the ε value, the higher the similarity with the path generated by the visibility graph (Figure 17c).



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

Figure 19. *Cont.*

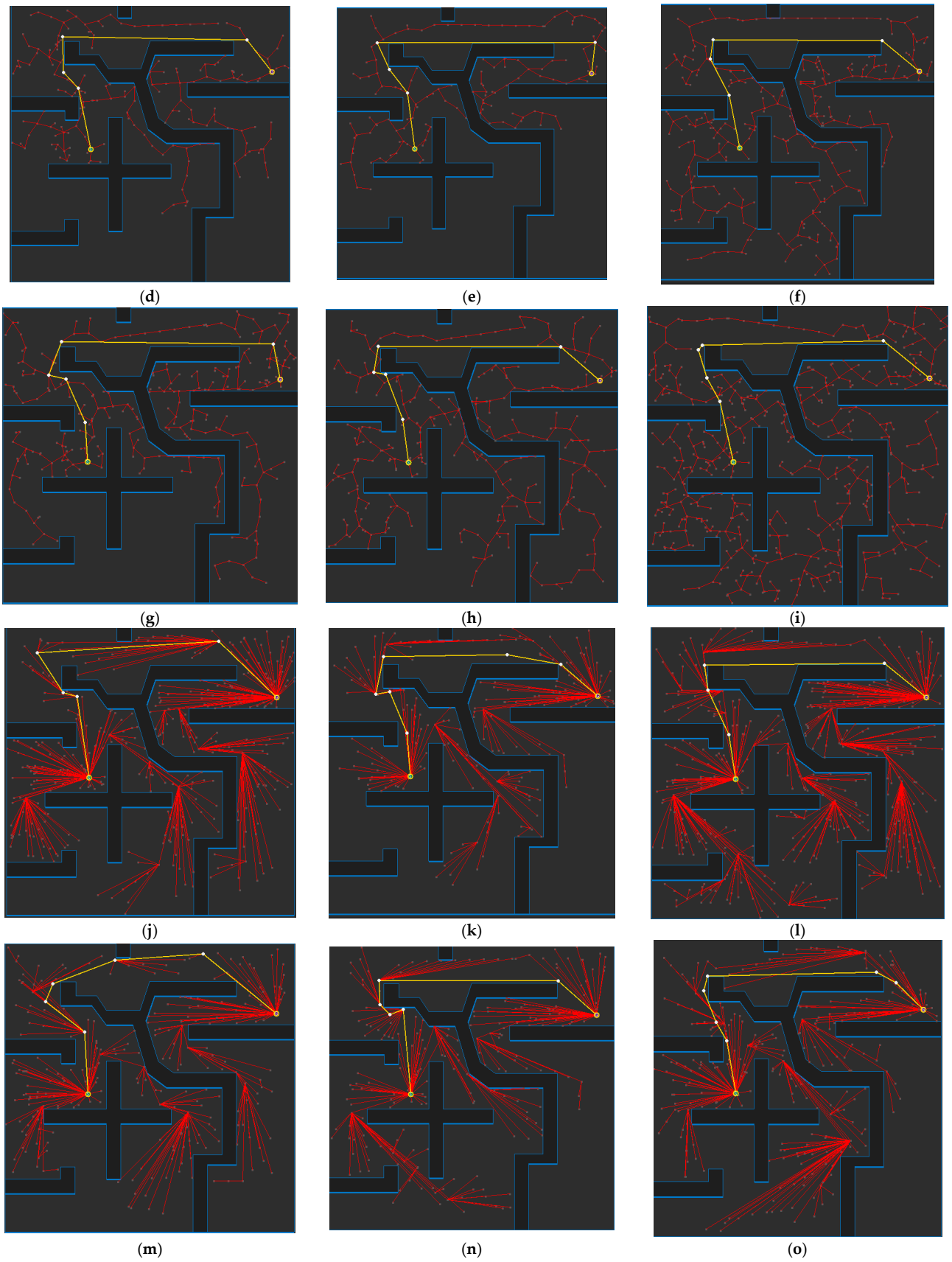


Figure 19. Experimental result of Map 3: (**a**) RRT-connect; (**b**) triangular-RRT-connect; (**c**) visibility graph; (**d**) RRT-connect

PTPMI (ε: 50 px); (**e**) RRT-connect PTPMI (ε: 30 px); (**f**) RRT-connect PTPMI (ε: 10 px); (**g**) RRT-connect bidirectional interpolation method (ε: 50 px); (**h**) RRT-connect bidirectional interpolation method (ε: 30 px); (**i**) RRT-connect bidirectional interpolation method (ε: 10 px); (**j**) triangular-RRT-connect PTPMI (ε: 50 px); (**k**) triangular-RRT-connect PTPMI (ε: 30 px); (**l**) triangular-RRT-connect PTPMI (ε: 10 px); (**m**) triangular-RRT-connect bidirectional interpolation method (ε: 50 px); (**n**) triangular-RRT-connect bidirectional interpolation method (ε: 30 px); (**o**) triangular-RRT-connect bidirectional interpolation method (ε: 10 px).

Table 6 summarizes the experimental results numerically for Map 3 based on RRT-connect. The case where the ε value of the proposed algorithm is 10 (px) results in the shortest path length compared to other cases and is closest to the path generated by the visibility graph (the relative ratio is 726/714, which is about 102%). It can be seen that the planning time of RRT-connect is shorter than that of PTPMI and BPTPMI. The biggest difference from RRT-Connect occurs when using the proposed algorithm with ε: 10 px. However, the difference between the values is very insignificant at 3 ms. Furthermore, the shortest planning time occurred when using the proposed algorithm with ε:50 px. At this time, compared to RRT-connect, the path was reduced by 218 px, and compared to PTPMI, it was reduced by 4 px. This means that for Map 3, the proposed algorithm guarantees optimality compared to other algorithms and has a similar planning time to basic RRT-connect.

Table 6. Experimental result based on RRT-connect of Map 3 (the parentheses to the right of each value of path length are relative ratios based on visibility graph (714 px)).

Performance	RRT-Connect	PTPMI Method			Bidirectional Interpolation Method		
		ε: 50 px	**ε: 30 px**	**ε: 10 px**	**ε: 50 px**	**ε: 30 px**	**ε: 10 px**
Path length (px)	1002 (140%)	788 (110%)	757 (106%)	729 (102%)	784 (110%)	755 (106%)	726 (102%)
Planning time (ms)	9	10	10	9	8	11	12

Table 7 summarizes the experimental results numerically for Map 3 based on triangular-RRT-connect. The case where the ε value of the proposed algorithm is ε: 10 px results in the shortest path length compared to other cases and is closest to the path generated by the visibility graph (the relative ratio is 727/714, which is about 102%). It can be seen that the planning time of triangular-RRT-connect is shorter than that of PTPMI and BPTPMI. The biggest difference from triangular-RRT-connect occurs when using the proposed algorithm. However, the difference between the values is very insignificant at 2 ms. Furthermore, the shortest path length occurred when using the proposed algorithm with ε: 10 px. At this time, compared to triangular-RRT-connect, the path was reduced by 86 px, and compared to PTPMI, it was reduced by 7 px. This means that for Map 3, the proposed algorithm guarantees optimality compared with other algorithms and has a similar planning time to basic RRT-connect.

Table 7. Experimental result based on triangular-RRT-connect of Map 3 (the parentheses to the right of each value of path length are relative ratios based on visibility graph (714 px)).

Performance	Triangular-RRT-Connect	PTPMI Method			Bidirectional Interpolation Method		
		ε: 50 px	**ε: 30 px**	**ε: 10 px**	**ε: 50 px**	**ε: 30 px**	**ε: 10 px**
Path length (px)	813 (114%)	787 (110%)	753 (105%)	729 (102%)	781 (109%)	750 (105%)	727 (102%)
Planning time (ms)	7	10	9	10	9	9	9

Figure 20 shows the path planning results for Map 4 for each algorithm. Looking at the generated path (yellow line), compared to Figure 20a, which is the result of RRT-connect, when PTPMI (Figure 20d–f) and the proposed algorithm (Figure 20g–i) were each applied for post-processing, the piecewise linear shape with sharp curves was reduced. In addition,

it can be seen that the smaller the ε value, the higher the similarity with the path generated by the visibility graph (Figure 18c).

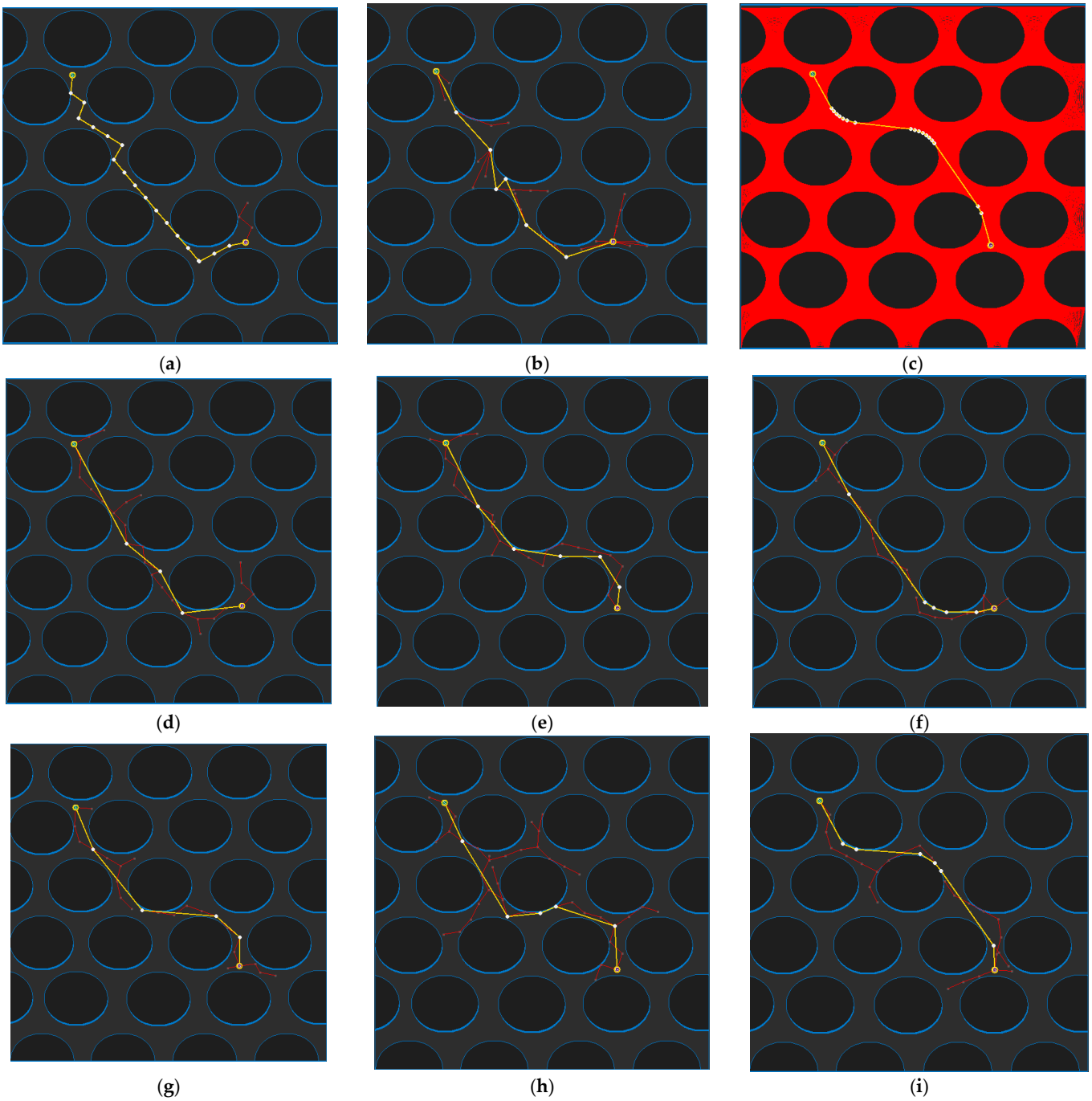


Figure 20. *Cont.*

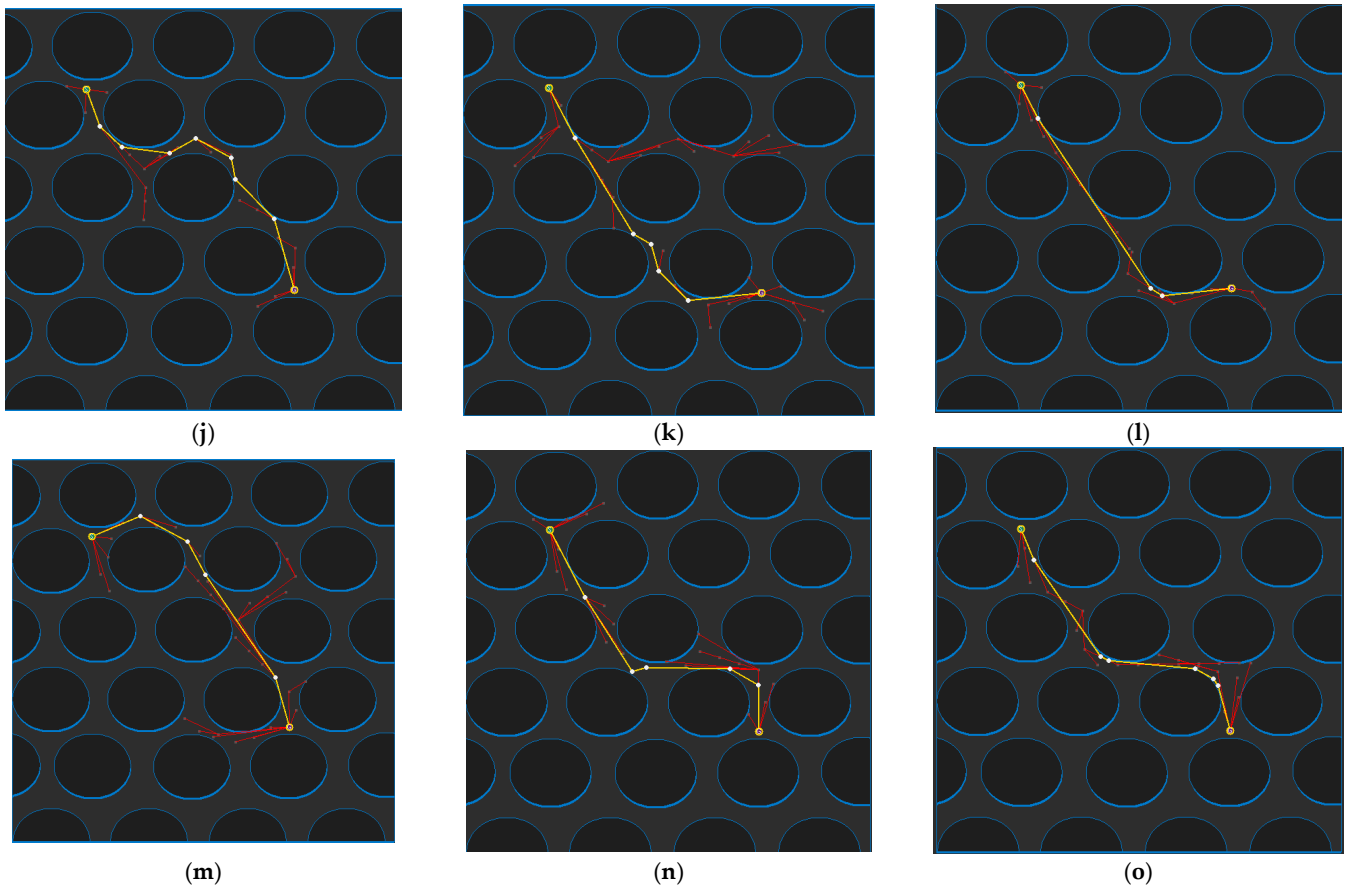


Figure 20. Experimental result of Map 4: (**a**) RRT-connect; (**b**) triangular-RRT-connect; (**c**) visibility graph; (**d**) RRT-connect PTPMI (ε: 50 px); (**e**) RRT-connect PTPMI (ε: 30 px); (**f**) RRT-connect PTPMI (ε: 10 px); (**g**) RRT-connect bidirectional interpolation method (ε: 50 px); (**h**) RRT-connect bidirectional interpolation method (ε: 30 px); (**i**) RRT-connect bidirectional interpolation method (ε: 10 px); (**j**) triangular-RRT-connect PTPMI (ε: 50 px); (**k**) triangular-RRT-connect PTPMI (ε: 30 px); (**l**) triangular-RRT-connect PTPMI (ε: 10 px); (**m**) triangular-RRT-connect bidirectional interpolation method (ε: 50 px); (**n**) triangular-RRT-connect bidirectional interpolation method (ε: 30 px); (**o**) triangular-RRT-connect bidirectional interpolation method (ε: 10 px).

Table 8 summarizes the experimental results numerically for Map 4 based on RRT-connect. It can be seen that the case where the ε value of the proposed algorithm is 10 (px) results in the shortest path length compared to other cases and is closest to the path generated by the visibility graph (relative ratio is 475/470, which is about 101%). It can be seen that the planning time of RRT-connect is shorter than that of PTPMI and bidirectional interpolation method. The biggest difference from RRT-connect occurs when using PTPMI with ε: 30 px. However, the difference between the values is very insignificant at 3 ms. Furthermore, the shortest planning time occurs when using the proposed algorithm with ε: 50 px. At this time, compared to RRT-connect, the path length is reduced by 72 px and has the same planning time as PTPMI. This means that the proposed algorithm for Map 4 guarantees optimality compared to other algorithms and has a similar planning time to basic RRT-connect.

Table 8. Experimental result based on RRT-connect of Map 4 (the parentheses to the right of each value of path length are relative ratios based on visibility graph (470 px)).

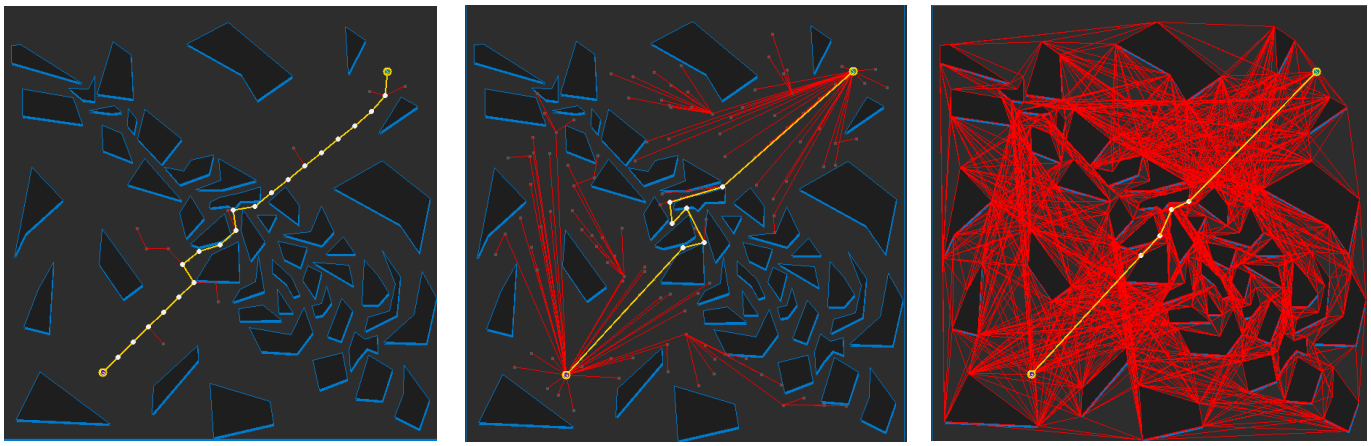
Performance	RRT-Connect	PTPMI Method			Bidirectional Interpolation Method		
		ε: 50 px	ε: 30 px	ε: 10 px	ε: 50 px	ε: 30 px	ε: 10 px
Path length (px)	576 (122%)	506 (108%)	496 (105%)	488 (104%)	504 (107%)	494 (105%)	475 (101%)
Planning time (ms)	1	2	4	2	1	3	2

Table 9 summarizes the experimental results numerically for Map 4 based on triangular-RRT-connect. It can be seen that the case where the ε value of the proposed algorithm is 10 (px) results in the shortest path length compared to other cases and is closest to the path generated by the visibility graph (relative ratio is 479/470, which is about 102%). Planning time is expressed as 1–2 ms in all cases. The biggest difference from triangular-RRT-connect occurs when using proposed algorithm with ε: 10 px. However, the difference between the values is very insignificant at 1 ms. This means that the proposed algorithm for Map 4 guarantees optimality compared to other algorithms and has a similar planning time to basic triangular-RRT-connect.

Table 9. Experimental result based on triangular-RRT-connect of Map 4 (the parentheses to the right of each value of path length are relative ratios based on visibility graph (470 px)).

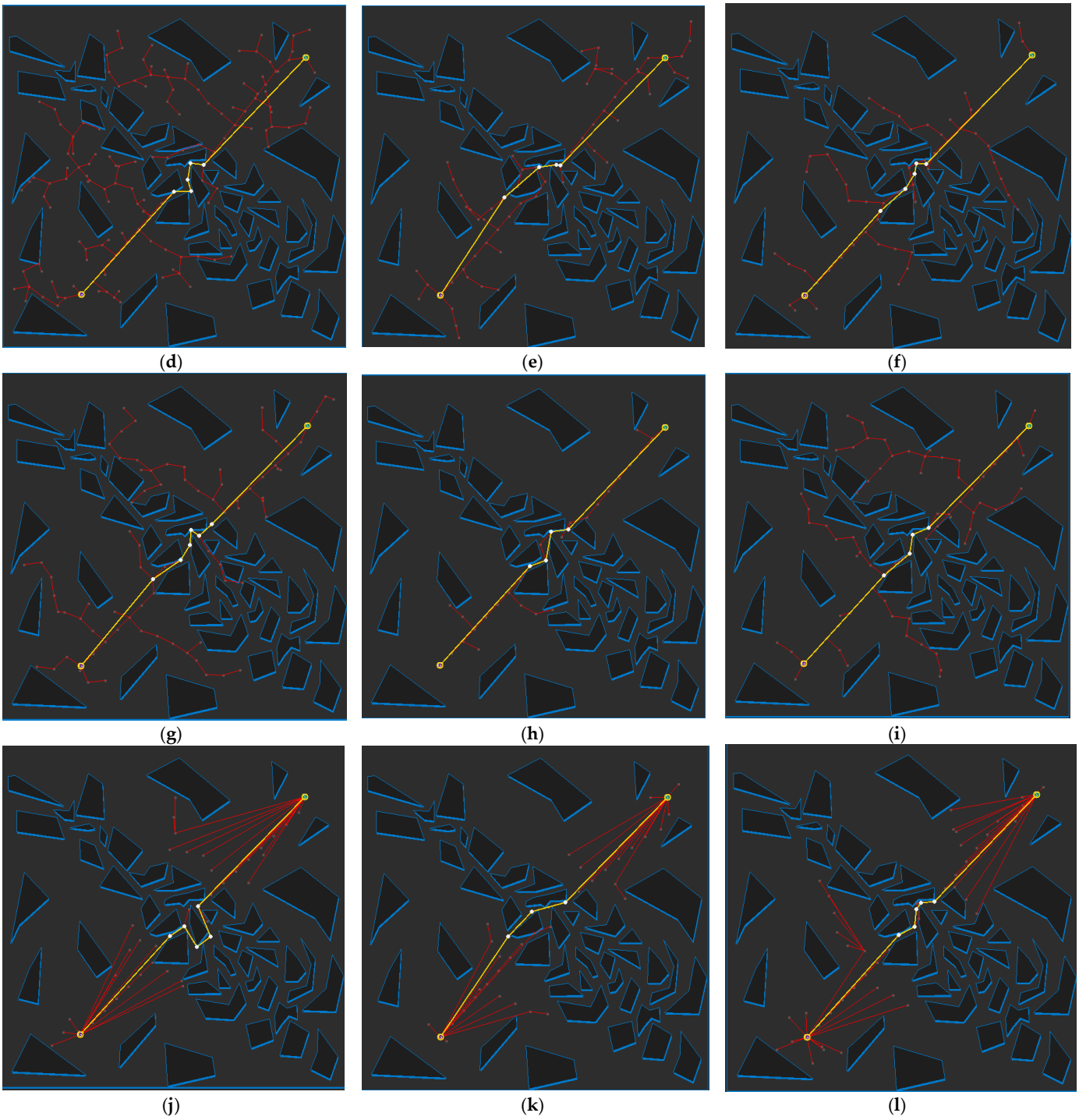
Performance	TriangularRRT-Connect	PTPMI Method			Bidirectional Interpolation Method		
		ε: 50 px	ε: 30 px	ε: 10 px	ε: 50 px	ε: 30 px	ε: 10 px
Path length (px)	514 (109%)	508 (108%)	499 (106%)	480 (102%)	514 (109%)	499 (106%)	479 (102%)
Planning time (ms)	1	1	1	2	1	2	2

Figure 21 shows the path planning results for Map 5 for each algorithm. Looking at the generated path (yellow line), compared to Figure 21a, which is the result of RRT-connect, when PTPMI (Figure 21d–f) and the proposed algorithm (Figure 21g–i) were each applied for post-processing, the piecewise linear shape with sharp curves was reduced. In addition, it can be seen that the smaller the ε value, the higher the similarity with the path generated by the visibility graph (Figure 19c).



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

Figure 21. *Cont.*

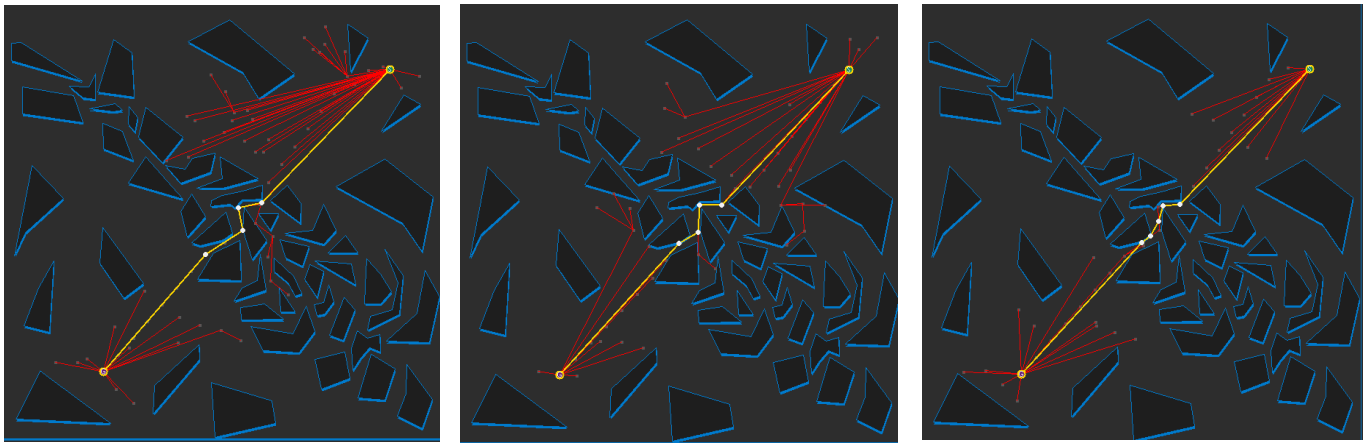


(**d**) (**e**) (**f**)

(**g**) (**h**) (**i**)

(**j**) (**k**) (**l**)

Figure 21. *Cont.*



(**m**) (**n**) (**o**)

Figure 21. Experimental result of Map 5: (**a**) RRT-connect; (**b**) triangular-RRT-connect; (**c**) visibility graph; (**d**) RRT-connect PTPMI (ε: 50 px); (**e**) RRT-connect PTPMI (ε: 30 px); (**f**) RRT-connect PTPMI (ε: 10 px); (**g**) RRT-connect bidirectional interpolation method (ε: 50 px); (**h**) RRT-connect bidirectional interpolation method (ε: 30 px); (**i**) RRT-connect bidirectional interpolation method (ε: 10 px); (**j**) triangular-RRT-connect PTPMI (ε: 50 px); (**k**) triangular-RRT-connect PTPMI (ε: 30 px); (**l**) triangular-RRT-connect PTPMI (ε: 10 px); (**m**) triangular-RRT-connect bidirectional interpolation method (ε: 50 px); (**n**) triangular-RRT-connect bidirectional interpolation method (ε: 30 px); (**o**) triangular-RRT-connect bidirectional interpolation method (ε: 10 px).

Table 10 summarizes the experimental results numerically for Map 5 based on RRT-connect. It can be seen that the case where the ε value of the proposed algorithm is 10 (px) results in the shortest path length compared to other cases, and the path is closest to the one generated by the visibility graph (the relative ratio is 646/576, which is about 112%). It can be seen that the planning time of RRT-connect is shorter than that of PTPMI and bidirectional interpolation method. The biggest difference from RRT-connect occurs when using PTPMI with ε: 10 px. However, the difference between the values is very insignificant at 3 ms. In other situations, it can be confirmed that the planned time is always 2 ms. This means that the proposed algorithm guarantees optimality for traversing Map 5 compared to other algorithms and the planning time differs from RRT-connect by 1 ms.

Table 10. Experimental result based on RRT-connect of Map 5 (the parentheses to the right of each value of path length are relative ratios based on visibility graph (576 px)).

Performance	RRT-Connect	PTPMI Method			Bidirectional Interpolation Method		
		ε: 50 px	**ε: 30 px**	**ε: 10 px**	**ε: 50 px**	**ε: 30 px**	**ε: 10 px**
Path length (px)	759 (132%)	689 (120%)	657 (114%)	653 (113%)	666 (117%)	655 (114%)	646 (112%)
Planning time (ms)	1	2	2	4	2	2	2

Table 11 summarizes the experimental results numerically for Map 5 based on triangular-RRT-connect. It can be seen that the case where the ε value of the proposed algorithm is 10 (px) results in the shortest path length compared to other cases, and the path is closest to the one generated by the visibility graph (the relative ratio is 641/576, which is about 111%). Planning time is expressed as 1–2 ms in all cases. The biggest difference from triangular-RRT-connect occurs when using proposed algorithm with ε: 50 px. However, the difference between the values is very insignificant at 1 ms. This means that the proposed algorithm guarantees optimality compared with other algorithms and the planning time differs from triangular-RRT-connect by 1 ms.

Table 11. Experimental result based on triangular-RRT-connect of Map 5 (the parentheses to the right of each value of path length are relative ratios based on visibility graph (576 px)).

Performance	Triangular-RRT-Connect	PTPMI Method			Bidirectional Interpolation Method		
		ε: 50 px	ε: 30 px	ε: 10 px	ε: 50 px	ε: 30 px	ε: 10 px
Path length (px)	673 (117%)	669 (116%)	666 (116%)	665 (115%)	663 (115%)	658 (114%)	641 (111%)
Planning time (ms)	1	1	1	2	2	1	1

Figure 22 shows the path planning results for Map 6 for each algorithm. Looking at the generated path (yellow line), compared to Figure 22a, which is the result of RRT-connect, when PTPMI (Figure 22d–f) and the proposed algorithm (Figure 22g–i) were each applied for post-processing, the piecewise linear shape with sharp curves was reduced. In addition, it can be seen that the smaller the ε value, the higher the similarity with the path generated by the visibility graph (Figure 20c).

Table 12 summarizes the experimental results numerically for Map 6 based on RRT-connect. The case where the ε value of the proposed algorithm is ε: 10 px results in the shortest path length compared to other cases and is closest to the visibility graph (the relative ratio is 1187/1165, which is about 101%). It can be seen that the planning time of RRT-Connect is shorter than that of PTPMI and the bidirectional interpolation method. The biggest difference from RRT-connect occurs when ε: 50 px of the algorithm is to be bounded. However, the difference between the values is very insignificant at 6 ms. Furthermore, the shortest planning time occurred when using PTPMI with ε: 50 px. This means that the proposed algorithm guarantees optimality compared to other algorithms, but takes an average of 4 ms longer for Map 6.

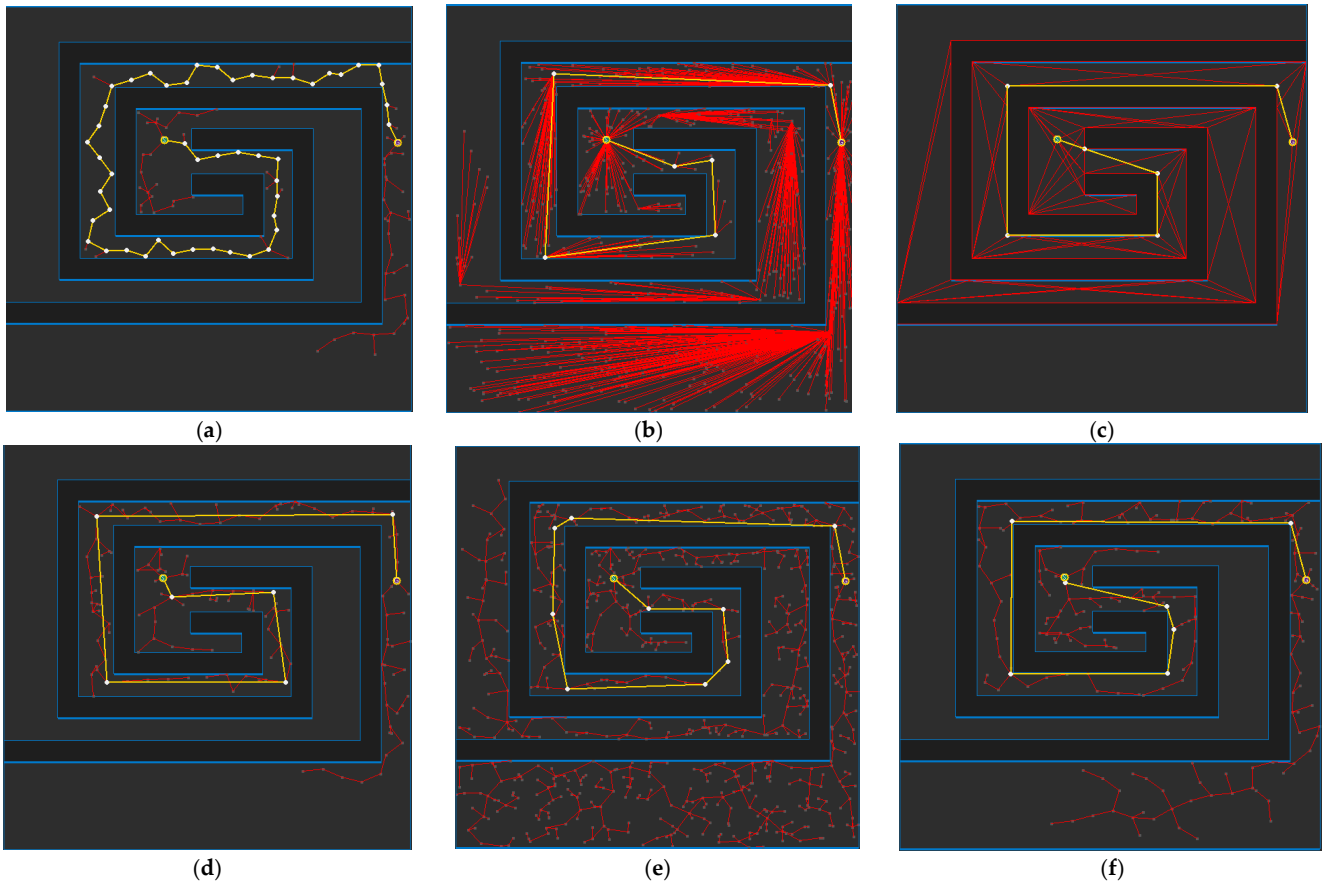


Figure 22. *Cont.*

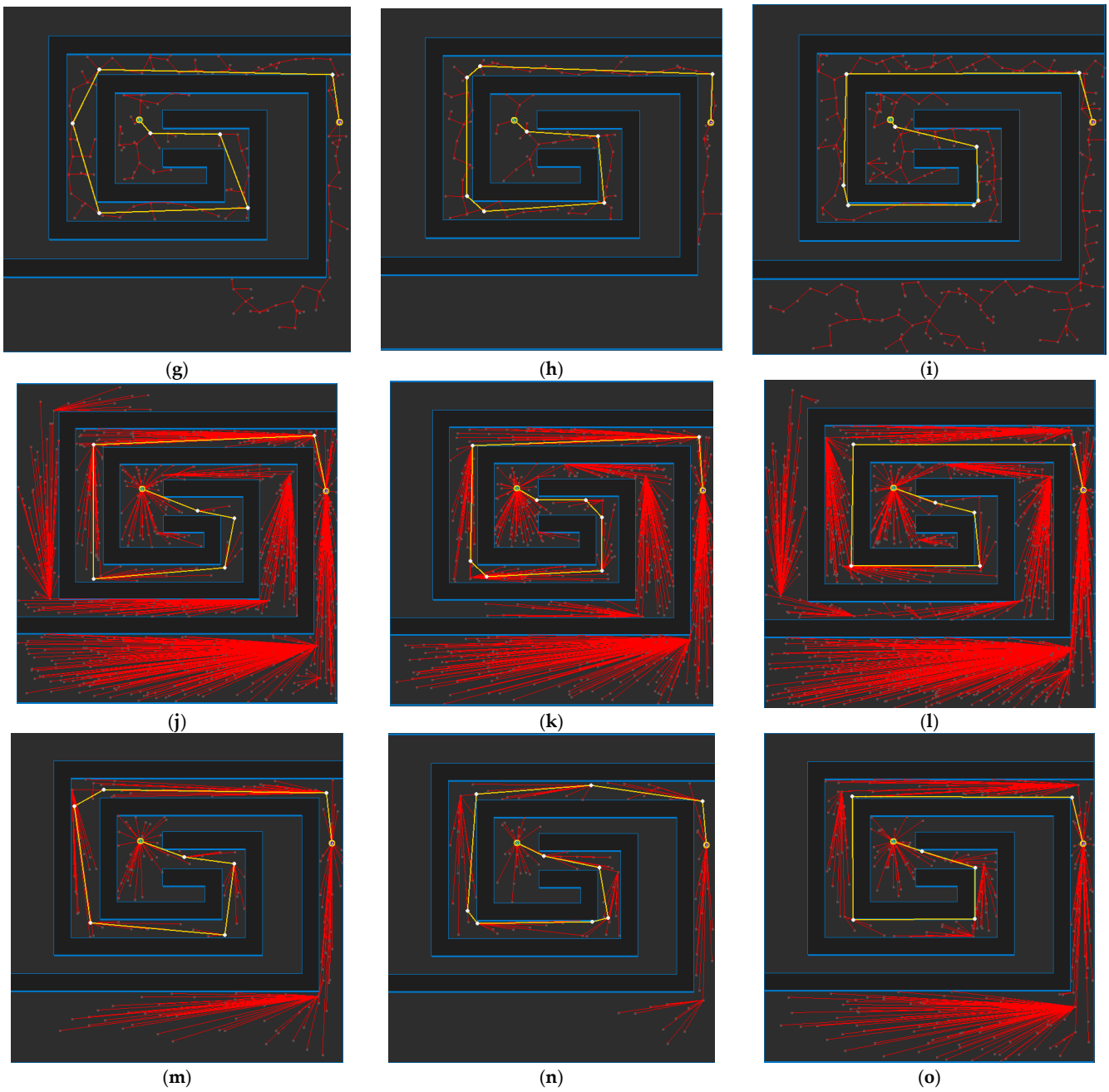


Figure 22. Experimental result of Map 6: (**a**) RRT-connect; (**b**) triangular-RRT-connect; (**c**) visibility graph; (**d**) RRT-connect PTPMI (ε: 50 px); (**e**) RRT-connect PTPMI (ε: 30 px); (**f**) RRT-connect PTPMI (ε: 10 px); (**g**) RRT-connect bidirectional interpolation method (ε: 50 px); (**h**) RRT-connect bidirectional interpolation method (ε: 30 px); (**i**) RRT-connect bidirectional interpolation method (ε: 10 px); (**j**) triangular-RRT-connect PTPMI (ε: 50 px); (**k**) triangular-RRT-connect PTPMI (ε: 30 px); (**l**) triangular-RRT-connect PTPMI (ε: 10 px); (**m**) triangular-RRT-connect bidirectional interpolation method (ε: 50 px); (**n**) triangular-RRT-connect bidirectional interpolation method (ε: 30 px); (**o**) triangular-RRT-connect bidirectional interpolation method (ε: 10 px).

Table 12. Experimental result based on RRT-connect of Map 6 (the parentheses to the right of each value of path length are relative ratios based on visibility graph (1165 px)).

Performance	RRT-Connect	PTPMI Method			Bidirectional Interpolation Method		
		ε: 50 px	ε: 30 px	ε: 10 px	ε: 50 px	ε: 30 px	ε: 10 px
Path length (px)	1478 (127%)	1292 (111%)	1257 (108%)	1189 (102%)	1286 (110%)	1254 (108%)	1187 (102%)
Planning time (ms)	24	24	27	26	30	26	28

Table 13 summarizes the experimental results numerically for Map 6 based on triangular RRT-connect. The case where the ε value of the proposed algorithm is ε: 10 px results in the shortest path length compared to other cases and is closest to the visibility graph (the relative ratio is 1186/1165, which is about 102%). The planning time is similar to triangular-RRT-connect with the proposed method. The biggest difference from triangular-RRT-connect occurs when ε: 50 px of the algorithm is to be bounded. However, the difference between the values is very insignificant at 1 ms. Furthermore, the shortest planning time occurred when using PTPMI with ε: 30 px. This means that the proposed algorithm guarantees optimality compared with other algorithms.

Table 13. Experimental result based on triangular-RRT-connect of Map 6 (the parentheses to the right of each value of path length are relative ratios based on visibility graph (1165 px)).

Performance	RRT-Connect	PTPMI Method			Bidirectional Interpolation Method		
		ε: 50 px	ε: 30 px	ε: 10 px	ε: 50 px	ε: 30 px	ε: 10 px
Path length (px)	1293 (111%)	1282 (110%)	1253 (108%)	1189 (102%)	1279 (110%)	1250 (107%)	1186 (102%)
Planning time (ms)	23	24	23	22	23	22	23

4.3. Experimental Results and Analysis

In this section, the experimental results of Maps 1 to 6 are summarized.

Table 14 is a table summarizing the experimental results on the path length. It can be seen that, for all maps, the proposed algorithm creates a shorter path compared to RRT-connect. The RRT-connect algorithm generates a path whose length is approximately 138% ((150 + 157 + 140 + 122 + 132 + 127)/6) longer on average compared to the visibility graph. Similarly, PTPMI generated about 113% longer paths with ε: 50 px, about 108% with ε: 30 px and about 105% longer with ε: 10 px compared to the visibility graph. In the case of the proposed algorithm, it can be seen that the path generated on average is about 112% longer with ε: 50 px, about 108% with ε: 30 px and about 104% longer at ε: 10 px compared to the visibility graph. Thus, the proposed algorithm has a path length closer to the visibility graph as the value of epsilon decreases. Based on the ε: 10 px of the proposed algorithm, the average path length decreased by about 34% compared to RRT-connect, and it was improved by about 1% compared to PTPMI.

Table 14. Total experimental result for path length based on RRT-connect (the parentheses to the right of each value of path length are relative ratios based on the visibility graph) (unit: px).

Map No.	RRT-Connect	PTPMI Method			Bidirectional Interpolation Method			Visibility Graph
		ε: 50 px	ε: 30 px	ε: 10 px	ε: 50 px	ε: 30 px	ε: 10 px	
Map 1	379 (150%)	283 (112%)	264 (104%)	258 (102%)	278 (110%)	263 (104%)	257 (101%)	253
Map 2	1843 (157%)	1399 (119%)	1326 (113%)	1230 (105%)	1395 (119%)	1324 (112%)	1223 (104%)	1172
Map 3	1002 (140%)	788 (110%)	757 (106%)	729 (102%)	784 (110%)	755 (106%)	726 (102%)	714
Map 4	576 (122%)	506 (108%)	496 (105%)	488 (104%)	504 (107%)	494 (105%)	475 (101%)	470
Map 5	759 (132%)	689 (120%)	657 (114%)	653 (113%)	666 (117%)	655 (114%)	646 (112%)	576
Map 6	1478 (127%)	1292 (111%)	1257 (108%)	1189 (102%)	1286 (110%)	1254 (108%)	1187 (102%)	1165

Table 15 is a table summarizing the experimental results on the path length based on triangular-RRT-connect. It can be seen that, for all maps, the proposed algorithm creates a shorter path compared to triangular-RRT-connect. The triangular-RRT-connect algorithm generates a path whose length is approximately 115% ((111 + 126 + 114 + 109 + 117 + 111)/6) longer on average compared to the visibility graph. Similarly, PTPMI generated about 112% longer paths with ε: 50 px, about 108% with ε: 30 px and about 105% longer with ε: 10 px compared to the visibility graph. In the case of the proposed algorithm, it can be seen that the path generated on average is about 112% longer with ε: 50 px, about 108% with ε: 30 px and about 104% longer at ε: 10 px compared to the visibility graph. Thus, the proposed algorithm has a path length a little closer to the visibility graph as the value of epsilon decreases. Based on the ε: 10 px of the proposed algorithm, the average path length decreased by about 11% compared to triangular-RRT-connect, and it was improved by about 1% compared to PTPMI.

Table 15. Total experimental result for path length based on triangular-RRT-connect (the parentheses to the right of each value of path length are relative ratios based on the visibility graph) (unit: px).

Map No.	Triangular-RRT-Connect	PTPMI Method			Bidirectional Interpolation Method			Visibility Graph
		ε: 50 px	ε: 30 px	ε: 10 px	ε: 50 px	ε: 30 px	ε: 10 px	
Map 1	282 (111%)	277 (109%)	264 (104%)	257 (101%)	274 (108%)	264 (104%)	257 (101%)	253
Map 2	1478 (126%)	1405 (120%)	1331 (113%)	1230 (105%)	1404 (120%)	1331 (113%)	1229 (105%)	1172
Map 3	813 (114%)	787 (110%)	753 (105%)	729 (102%)	781 (109%)	750 (105%)	727 (102%)	714
Map 4	514 (109%)	508(108%)	499(106%)	480(102%)	514(109%)	499(106%)	479(102%)	470
Map 5	673 (117%)	669 (116%)	666 (116%)	665 (115%)	663 (115%)	658 (114%)	641 (111%)	576
Map 6	1293 (111%)	1282 (110%)	1253 (108%)	1189 (102%)	1279 (110%)	1250 (107%)	1186 (102%)	1165

Table 16 summarizes the experimental results on the planning time. In all maps, it can be seen that the proposed algorithm takes longer than RRT-connect. However, the difference is not large. Based on ε: 10 px, which was confirmed to be closest to optimality through Table 13, the biggest difference with RRT-connect, of 4 ms, occurs for Map 6.

Table 16. Experimental result for planning time based on RRT-connect (unit: ms).

Map No.	RRT-Connect	PTPMI Method			Bidirectional Interpolation Method		
		ε: 50 px	ε: 30 px	ε: 10 px	ε: 50 px	ε: 30 px	ε: 10 px
Map 1	<0	<0	<0	<0	1	<0	<0
Map 2	220	242	264	250	243	272	223
Map 3	9	10	10	9	8	11	12
Map 4	1	2	4	2	1	3	2
Map 5	1	2	2	4	2	2	2
Map 6	24	24	27	26	30	26	28

As the proposed algorithm comprises an additional procedure to approach obstacles compared to PTPMI, it is predicted that it would require more planning time compared to PTPMI. The case in which the proposed algorithm takes the most time compared to PTPMI is when the ε value in Map 2 is 30 px, and the difference between the two is 8 ms. However, there are cases where the time of the proposed algorithm is reduced compared to PTPMI. In particular, for Map 2, when ε: 10 px, the proposed algorithm requires 223 ms, whereas PTPMI requires 250 ms, which is a reduction of 27 ms for the proposed algorithm. This may be due to the reason that the planning time of the proposed algorithm is more affected by the random sampling effects, an intrinsic problem of the RRT-series algorithm, than by the time required to process the additional procedure.

Table 17 summarizes the experimental results on the planning time. It can be seen that the proposed algorithm takes longer than triangular-RRT-connect in most maps. However, the difference is not large. Based on ε: 10 px, which was confirmed to be closest to optimality

through Table 14, the biggest difference with RRT-Connect, of 2 ms, occurs for Map 3. Even in Maps 2 and 6, it can be seen that the time is reduced. %clearpage

Table 17. Experimental result for planning time based on triangular-RRT-connect (unit: ms).

Map No.	TriangularRRT-Connect	PTPMI Method			Bidirectional Interpolation Method		
		ε: 50 px	ε: 30 px	ε: 10 px	ε: 50 px	ε: 30 px	ε: 10 px
Map 1	<0	<0	<0	<0	<0	<0	<0
Map 2	195	197	194	214	205	181	194
Map 3	7	10	9	10	9	9	9
Map 4	1	1	1	2	1	2	2
Map 5	1	1	1	2	2	1	1
Map 6	27	24	20	22	19	22	23

As the proposed algorithm comprises an additional procedure to approach obstacles compared to PTPMI, it is predicted that it would require more planning time compared to PTPMI. The case in which the proposed algorithm takes the most time compared to PTPMI is when the ε value in Map 2 is 50 px, and the difference between the two is 8 ms. However, there are cases where the time of the proposed algorithm is reduced compared to PTPMI. In particular, for Map 2, when ε: 30 px, the proposed algorithm requires 181 ms, whereas PTPMI requires 194 ms, which is a reduction of 13 ms for the proposed algorithm. Thus, it can be confirmed that the planning time of the proposed algorithm is more affected by the probabilistic integrity, an intrinsic problem of the RRT-like algorithms, than by the time required to process the additional procedure.

Figure 23 shows the overall result for Map 1. The rectangle represents worst path length, the circle represents average path length and the triangle represents best path length. First of all, it can be seen that the results similar to the visibility graph appear in all cases except for RRT-connect in best path length (triangle). Average path length (circle) is shorter when the post-processing method (PTPMI or proposed method) is applied than the original RRT-connect and triangular-RRT-connect. It can be seen that worst path length (rectangle) gradually approaches average as the ε value decreases in the post-processing method. In other words, if the post-processing method is applied, worst path length (rectangle) is enhanced. Moreover, the proposed method enhanced worst path length better than PTPMI.

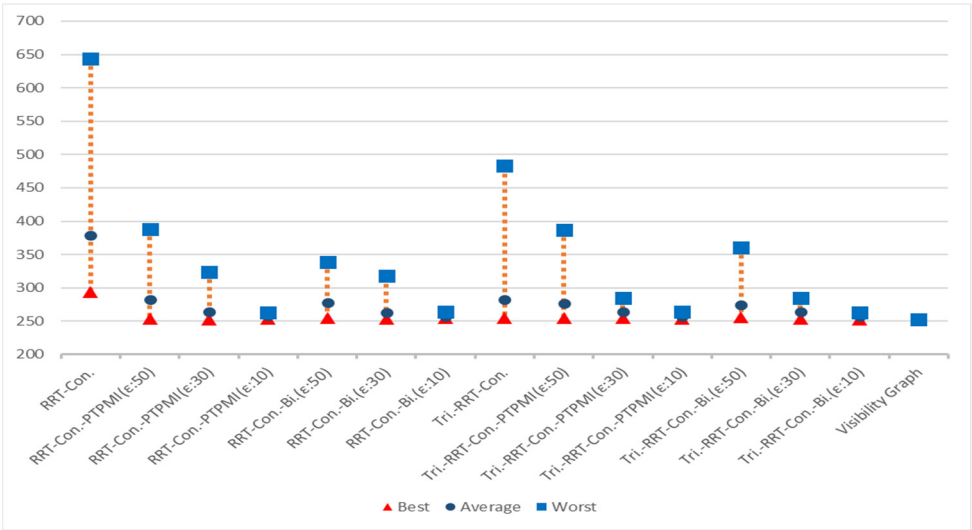


Figure 23. Best, average and worst path length by algorithm in Map 1.

Figure 24 shows the overall result for Map 2. It can be seen that the path length is improved in all cases (best, worst, average) when the post-processing method is applied

rather than the original algorithm. In addition, as the ε value decreased in the post-processing method, the path length was enhanced in all cases (best, worst, average). If the ε value is the same, the proposed method is more enhanced than the PTPMI in all cases (best, worst, average).

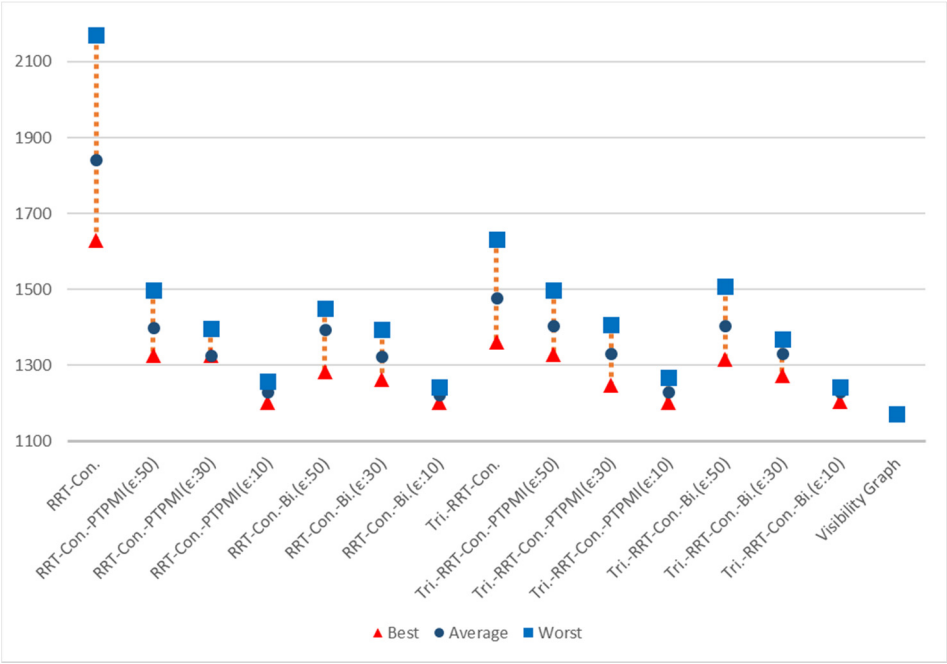


Figure 24. Best, average and worst path length by algorithm in Map 2.

Figure 25 shows the overall result for Map 3. It can be seen that the path length is improved in all cases (best, worst, average) when the post-processing method is applied rather than the original algorithm. In addition, as the ε value decreased in the post-processing method, the path length was enhanced in all cases (best, worst, average). If the ε value is the same, the proposed method is more enhanced than the PTPMI in all cases (best, worst, average).

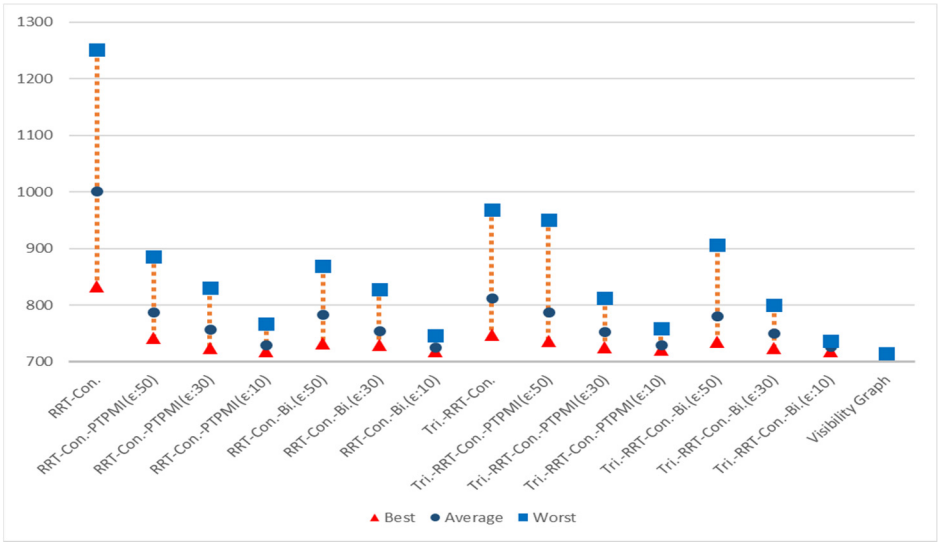


Figure 25. Best, average and worst path length by algorithm in Map 3.

Figure 26 shows the overall result for Map 4. It can be seen that the results similar to the visibility graph appear in all cases except for RRT-connect in best path length (triangle). When the post-processing method is applied, if ε value is reduced, average and worst are enhanced. Moreover, if ε value is the same in the post-processing method, the proposed method enhanced worst more than PTPMI.

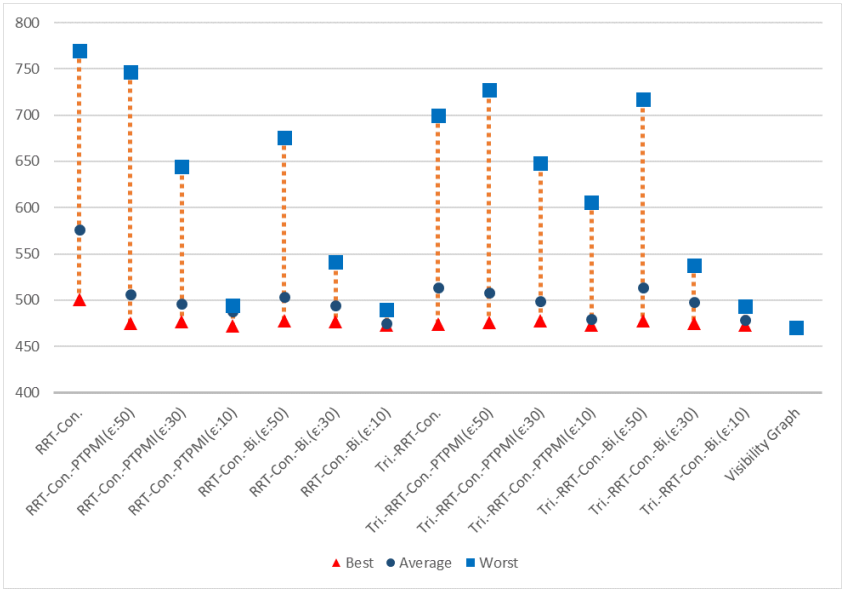


Figure 26. Best, average and worst path length by algorithm in Map 4.

Figure 27 shows the overall result for Map 5. It can be seen that the results similar to the visibility graph appear in all cases. When the post-processing method is applied, if ε value is reduced, worst is enhanced. Moreover, if ε value is the same in the post-processing method, the proposed method improves worst more than PTPMI.

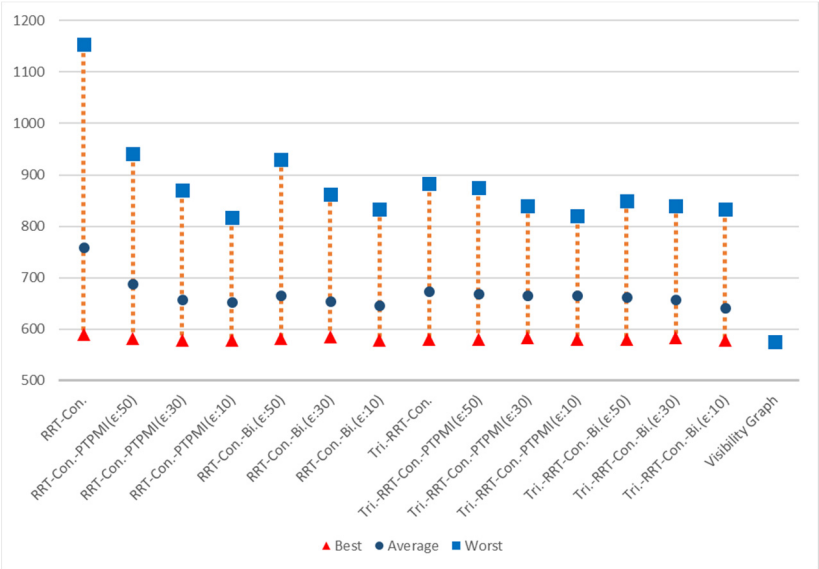


Figure 27. Best, average and worst path length by algorithm in Map 5.

Figure 28 shows the overall result for Map 6. It can be seen that the path length is improved in all cases (best, worst, average) when the post-processing method is applied rather than the original algorithm. In addition, as the ε value reduced in the post-processing method, the path length was enhanced in all cases (best, worst, average). If the ε value

is the same, the proposed method is more enhanced than the PTPMI in all cases (best, worst, average).

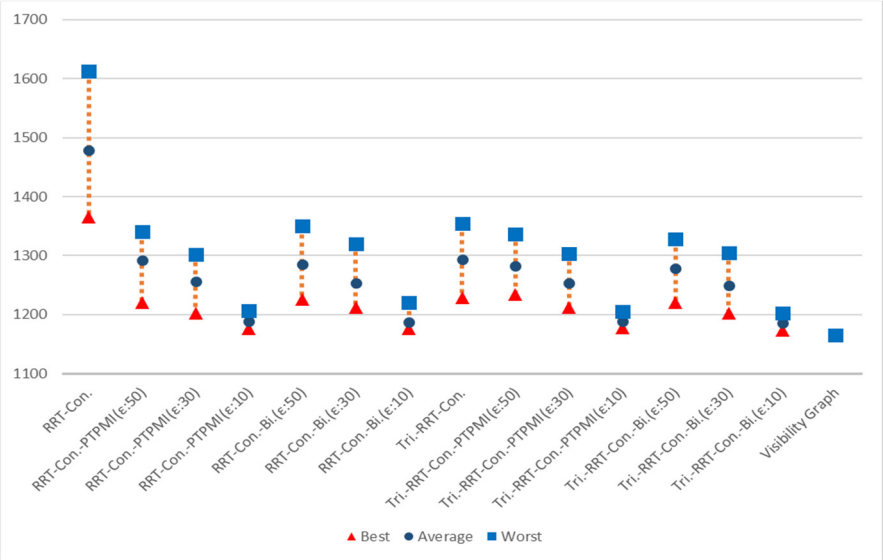


Figure 28. Best, Average and Worst path length by algorithm in Map 6.

5. Conclusions

In this paper, we proposed the bidirectional interpolation method. The proposed method can minimize the planning time and overcome the limit of optimality of sampling-based algorithms and kinodynamic error.

It was confirmed that bidirectional interpolation method plans a path close to the optimum when applied to the existing RRT-like algorithms through mathematical modeling. Simulations were performed to confirm the performance of bidirectional interpolation method. In six different environment maps, it was confirmed that the path length was shortened by 34% on average compared to when the basic RRT-connect algorithm was applied, and the path length was, on average, only 4% longer than the visibility graph. In addition, bidirectional interpolation method has the advantage of being applicable to all path planning methods that plan a locally piecewise linear path.

Compared with the PTPMI algorithm, it was confirmed that the proposed method shows a little better but is staggered with performance by 1–3% in terms of path length, and there was no significant difference in terms of planning time.

In most cases with all maps, the proposed method shows that worst path length was greatly reduced when the post-processing method was applied. In addition, if the ε value is the same, worst path length of the proposed algorithm is improved over PTPMI.

In this paper, a method applicable to the RRT-likes is proposed. However, the proposed method is a technique that can be applied to sampling-based planning algorithms such as RRT. Therefore, it can be applied in various fields where motion planning of robots is used, such as mobile robots, manipulators and drones.

Author Contributions: Idea and conceptualization: J.-G.K., T.-W.K. and J.-W.J.; methodology: J.-G.K., T.-W.K. and J.-W.J.; software: T.-W.K., J.-G.K. and J.-W.J.; experiment: T.-W.K., J.-G.K. and J.-W.J.; validation: T.-W.K., J.-G.K. and J.-W.J.; investigation: T.-W.K., J.-G.K. and J.-W.J.; resources: J.-G.K. and J.-W.J.; writing: T.-W.K., J.-G.K. and J.-W.J.; visualization: T.-W.K., 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. Schwab, K. *The Fourth Industrial Revolution*; Currency: New York, NY, USA, 2017.
2. Marin-Plaza, P.; Hussein, A.; Martin, D.; Escalera, A.D.L. Global and local path planning study in a ROS-based research platform for autonomous vehicles. *J. Adv. Transp.* **2018**, *2018*, 6392697. [CrossRef]
3. Sariff, N.; Buniyamin, N. An overview of autonomous mobile robot path planning algorithms. In Proceedings of the IEEE 4th Student Conference on Research and Development, Selangor, Malaysia, 28–29 June 2006; pp. 183–188.
4. LaValle, S.M. *Rapidly-Exploring Random Trees: A New Tool for Path Planning*; Springer: London, UK, 1998.
5. Kang, J.-G.; Jung, J.-W. Post Triangular Rewiring Method for Shorter RRT Robot Path Planning. *Int. J. Fuzzy Log. Intell. Syst.* **2021**, *21*, 213–221. [CrossRef]
6. LaValle, S.M.; Kuffner, J.J., Jr. Randomized kinodynamic planning. *Int. J. Robot. Res.* **2001**, *20*, 378–400. [CrossRef]
7. Roy, D. Visibility graph based spatial path planning of robots using configuration space algorithms. *Robot. Auton. Syst.* **2009**, *24*, 1–9. [CrossRef]
8. Katevas, N.I.; Tzafestas, S.G.; Pnevmatikatos, C.G. The approximate cell decomposition with local node refinement global path planning method: Path nodes refinement and curve parametric interpolation. *J. Intell. Robot. Syst.* **1998**, *22*, 289–314. [CrossRef]
9. 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; pp. 316–321.
10. 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]
11. Kwon, J.; Choi, K. Kinodynamic Model Identification: A Unified Geometric Approach. *IEEE Trans. Robot.* **2021**, *37*, 1100–1114. [CrossRef]
12. Kurzer, K. Path Planning in Unstructured Environments: A Real-Time Hybrid A* Implementation for Fast and Deterministic Path Generation for the KTH Research Concept Vehicle. Master's Thesis, KTH Royal Institute of Technology School of Engineering Sciences, Stockholm, Sweden, 2016.
13. Buniyamin, N.; Ngah, W.W.; Sariff, N.; Mohamad, Z. A simple local path planning algorithm for autonomous mobile robots. *Int. J. Syst. Appl. Eng. Dev.* **2011**, *5*, 151–159.
14. Donald, B.; Xavier, P.; Canny, J.; Reif, J. Kinodynamic motion planning. *J. ACM* **1993**, *40*, 1048–1066. [CrossRef]
15. Wang, J.; Li, B.; Meng, M.Q.-H. Kinematic Constrained Bi-directional RRT with Efficient Branch Pruning for robot path planning. *Expert Syst. Appl.* **2021**, *170*, 114541. [CrossRef]
16. Karaman, S.; Frazzoli, E. Sampling-based algorithms for optimal motion planning. *Int. J. Robot. Res.* **2011**, *30*, 846–894. [CrossRef]
17. Kang, J.-G.; Choi, Y.-S.; Jung, J.-W. An Enhancement Method of Rapidly-exploring Random Tree Robot Path Planning using Midpoint Interpolation. *Appl. Sci.* **2021**, *11*, 8483. [CrossRef]
18. Craig, J.J. *Introduction to Robotics: Mechanics and Control*, 3rd ed.; Pearson Education: Delhi, India, 2009.
19. Dijkstra, E.W. A note on two problems in connexion with graphs. *Numer. Math.* **1959**, *1*, 269–271. [CrossRef]
20. 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]
21. Jung, J.-W.; Park, J.-S.; Kang, T.-W.; Kang, J.-G.; Kang, H.-W. Mobile Robot Path Planning Using a Laser Range Finder for Environments with Transparent Obstacles. *Appl. Sci.* **2020**, *10*, 2799. [CrossRef]
22. Geraerts, R.; Overmars, M.H. A comparative study of probabilistic roadmap planners. In *Algorithmic Foundations of Robotics V*; Springer: Berlin/Heidelberg, Germany, 2004; pp. 43–57.
23. 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; pp. 995–1001.
24. 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.
25. 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. [CrossRef] [PubMed]

26. 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.
27. Han, J. Mobile robot path planning with surrounding point set and path improvement. *Appl. Soft Comput.* **2017**, *57*, 35–47. [CrossRef]
28. Yoon, H.U.; Lee, D.-W. Subplanner algorithm to escape from local minima for artificial potential function based robotic path planning. *Int. J. Fuzzy Log. Intell. Syst.* **2018**, *18*, 263–275. [CrossRef]