OXFORD CDE

Journal of Computational Design and Engineering, 2024, **11**, 303–317

DOI: 10.1093/jcde/qwae085
Advance access publication date: 3 October 2024
Research Article

Automatic cable routing based on improved pathfinding algorithm and B-spline optimization for collision avoidance

Kunchan Kim[1], Yeongjun Yoon[1], Byung Chul Kim[2], Jongguk Kim[3], Soonhung Han[4] and Soonjo Kwon[5,*]

[1]Department of Mechanical Engineering, Graduate School, Kumoh National Institute of Technology, 61 Daehak-ro, Gumi 39177, Korea
[2]School of Mechanical Engineering, Korea University of Technology and Education, 1600 Chungjeol-ro, Cheonan 34036, Korea
[3]Elozen Co. Ltd, 205 Yangji-ro, Bucheon 14786, Korea
[4]Korea STEP Center, 314 Gyeryong-ro, Daejeon 30147, Korea
[5]School of Mechanical System Engineering, Kumoh National Institute of Technology, 61 Daehak-ro, Gumi 39177, Korea
*Correspondence: soonjo.kwon@kumoh.ac.kr

Abstract

With the recent growth of electrical and electronic systems such as electric vehicles, the demand for automatic cable routing for electrical wiring design is increasing. However, real industry use cases of automatic cable routing are still rare especially in three-dimensional design. In this study, we propose a new pathfinding algorithm, JPS–Theta*, which combines the existing pathfinding algorithms, Jump Point Search and Theta*, that is better suited for cable routing. In addition, we propose a B-spline optimization algorithm to create natural cable shapes while avoiding collisions. In the experiments, it was found that the proposed pathfinding algorithm complements the existing algorithms and is thought to be more suitable for the purpose of automatic cable routing. Additionally, ant colony optimization for continuous domains, a meta-heuristic algorithm, was successfully used for optimizing the B-spline to obtain cable shapes without collision. Lastly, as a case study, the proposed method was directly applied to the electrical panel design to show its effectiveness. We expect that the proposed method will be able to improve the efficiency and quality of electrical wiring design.

Keywords: automatic cable routing, pathfinding algorithm, Jump Point Search, Theta*, B-spline optimization, ant colony optimization

1. Introduction

In the electrical wiring design process, cable routing is one of the tasks that requires a lot of time in various industries such as vehicles, aviation, and construction. Similarly, it is known that pipe routing in the design of the plant industry accounts for 50% of the entire design process (Park & Storch, 2002). Recently, many systems that previously operated on fossil fuels are being substituted by systems using electricity. As demand for electrical and electronic systems increases, the demand for automatic cable routing systems is increasing. Fig. 1 shows examples of automatic cable routing applied in the automotive industry (Hermansson *et al.*, 2013, 2016; Kim *et al.*, 2021b). However, up until now, it is rare to see universal applications of automatic cable routing in 3D (three-dimensional) design (Kim *et al.*, 2021b).

The automatic cable routing problem is to find a cable path that satisfies constraints such as cost, obstacle avoidance, and natural cable shape in a 3D design space (Kim *et al.*, 2021a). To suggest the optimal route, existing research on automatic routing mainly used pathfinding algorithms and meta-heuristic algorithm. The existing pathfinding algorithms suggest an optimal path that does not pass through the obstacles, given the start and end points in a design space. Representative pathfinding algorithms include Dijkstra, A*, Theta*, and Jump Point Search (JPS) algorithms.

The Dijkstra algorithm has been widely adopted for routing tasks although it is relatively old (Dijkstra, 2022). The A* algorithm is used in various fields that utilize the optimal path. It further considers potential heuristic costs in Dijkstra's algorithm (Hart *et al.*, 1968). The JPS algorithm is a relatively recent algorithm that shows overwhelming performance compared to other algorithms in the grid environment and guarantees the optimal path (Harabor & Grastien, 2011). The Theta* algorithm is one of the variants of A* and returns a path without angle constraints by considering line of sight (LOS) during the node traversal. It excludes nodes that are not necessary for the final path (Daniel *et al.*, 2010).

Compared to pipe routing, cable routing path has relatively less restrictions on direction. In pathfinding algorithms, execution time and length of the path are important evaluation indicators. Therefore, they are considered in optimizing cable routing as well as reducing cable costs. In this study, we propose a new pathfinding algorithm that combines the JPS and Theta* algorithms to fully leverage the advantages of both to reduce execution time and the length of the path. This combination utilizes the characteristics of existing Theta* to eliminate unnecessary intermediate points, making it easier to optimize by reducing the complexity of the generated cable model, and shows a relatively fast execution time due to it is based on JPS.

Received: May 30, 2024. **Revised:** September 26, 2024. **Accepted:** September 27, 2024

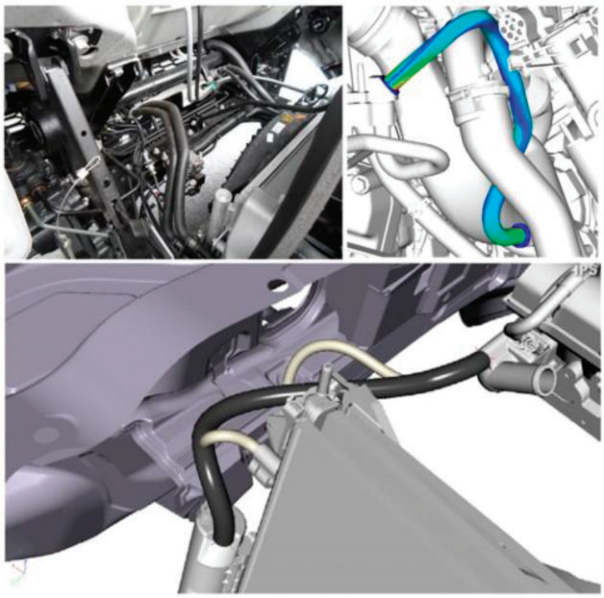

Figure 1: Examples of cable, harness, and hose routing.

In cable routing, it is important to present an optimized cable shape that satisfies design conditions, and this has been treated as an important issue in previous studies. In this study, interference, length, and minimum bend radius of the cable were used to design constraints and objective functions, considering stability, cost-effectiveness, and feasibility. Additionally, we propose a pre-processing step before optimization to achieve better optimization results with lower computational cost. We use random follower ACOR (RFACOR), derived from ant colony optimization for continuous domains (ACOR), a type of meta-heuristic algorithm to optimize the B-spline cable shape (Zhou *et al.*, 2023).

The main contributions of this paper can be summarized as follows.

(1) We proposed JPS–Theta*, a new pathfinding algorithm combining JPS and Theta* that excludes unnecessary intermediate interpolation points for a simplified and stable cable path.
(2) We propose a new cable shape optimization considering collision avoidance and verify the effectiveness of the proposed method through experiments.
(3) We presented a case in which JPS–Theta* and B-spline optimization were applied to actual electrical panel design to show the effectiveness of the proposed.

The overall process of the automatic cable routing proposed in this study is shown in Fig. 2. In the data import stage, the cable input/output terminals and design geometry information are imported in XML (eXtensible Markup Language) and ISO 10303 STEP (STandard for the Exchange of Product model data) formats, respectively. Then, in the pre-processing stage, point clouds and voxels are generated based on the design geometry to apply pathfinding and optimization algorithms. In the path planning stage, an initial B-spline shape passing through the initial path points is obtained using a pathfinding algorithm. Finally, in the cable optimization stage, cable shape optimization is performed to achieve a shape that satisfies the design conditions, i.e., collision avoidance.

This paper is structured as follows. First, Section 2 introduces related study. Section 3 introduces the A*, JPS, and Theta* algorithms, presents a new LOS-based JPS algorithm, JPS–Theta*, that combines the advantages of JPS and Theta*, and introduces experimental results to verify the effectiveness of the proposed pathfinding algorithm. In Section 4, we proposed a method to generate an optimized cable shape that satisfies constraints and design conditions such as maximum curvature, cable length, and obstacle collisions using RFACOR, the state-of-the-art

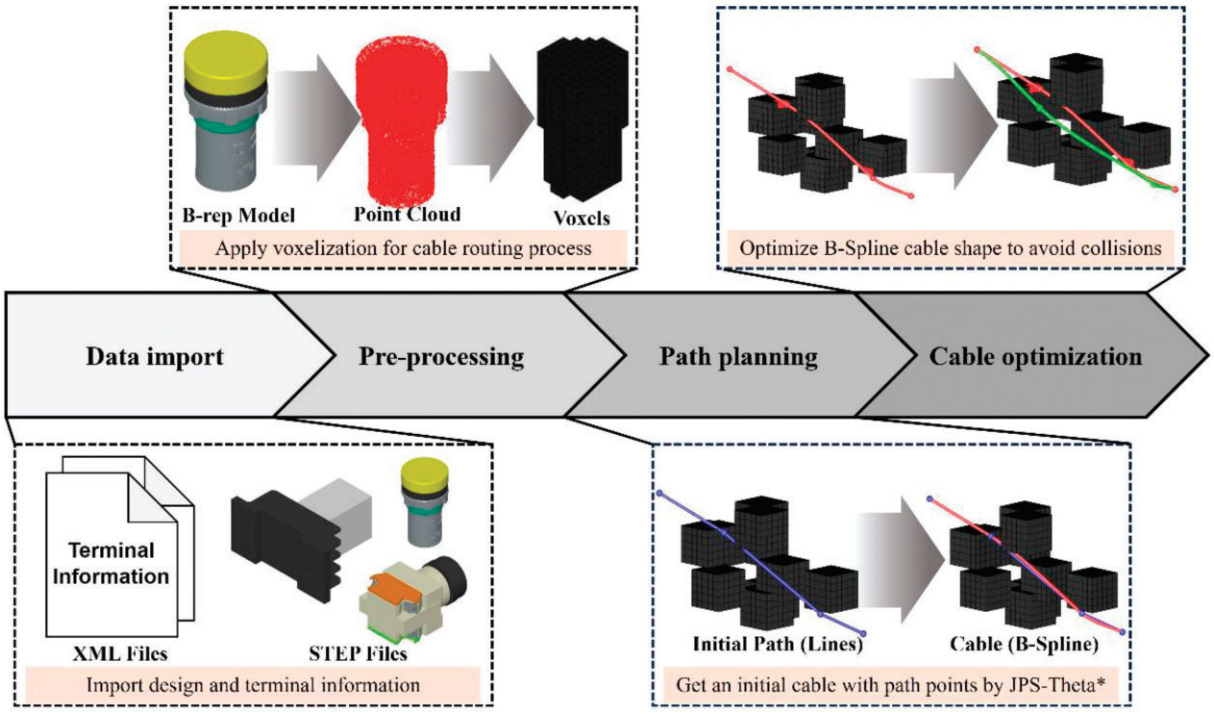

Figure 2: Overall process of the proposed automatic cable routing.

meta-heuristic algorithm, and introduces experimental results to verify the effectiveness of the cable optimization algorithm. Section 5 introduces a case study applied to actual electrical panel design. Finally, Section 6 concludes this study with future research directions.

2. Related work

In this section, we introduce related studies, categorized into domains of pipe and cable routing, pathfinding algorithms, and path smoothing algorithms.

2.1 Pipe and cable routing

In response to needs in numerous industries including piping and electrical design, automatic routing for pipes and cables has been developed. Similar algorithms are used with the goal of generating pipes or cables that travel around obstacles given start and end points, while specific requirements differ by field. The algorithms used for routing can be broadly divided into two types. First, there are studies that search for the optimal routing path within a graph or network structure. Second, there are studies proposing the optimal routing path within a 2D (two-dimensional) or 3D grid environment (Kim *et al.*, 2021a).

In studies where the design space is represented as a graph, the graph analysis is typically conducted first, followed by searching for the optimal route considering the constraints of the graph nodes and edges. There was a study that interpreted a cooling system as a graph and performed optimized routing using the genetic algorithm (Chan *et al.*, 2007). Another study represented circumferential spaces, such as an aero engine, as a graph to obtain the optimal route (Liu & Wang, 2015). More recently, a study has been presented proposing a multi-cable routing algorithm in a graph space for automatic cable routing in commercial vehicles like trucks (Kim *et al.*, 2021a,b).

Research employing 2D or 3D grids for routing have been vigorously conducted, with many instances stemming from the shipbuilding sector. In the shipbuilding domain, there was a study that categorized obstacles in the areas where pipes should pass as special obstacle nodes, introducing an algorithm that leads to natural obstacle avoidance (Kang *et al.*, 1999; Fan *et al.*, 2006). Another study applied the JPS for ship pipe auto-routing. In this study, a modified JPS was presented in line with pipe design constraints such as reducing the number of elbows, excluding diagonal routes, and setting preferred areas (Min *et al.*, 2020). Recently, research has been conducted applying curriculum-based reinforcement learning for ship pipe routing (Kim *et al.*, 2023). Additionally, there have been studies conducted to further optimize the calculated routing path. In the routing design process of electrical wiring interconnection system, there was a study that dealt with additional global optimization in the initial local optimization-based path (Zhu *et al.*, 2017). Additionally, a method to solve the cable routing problem defined as a Bi-optimization problem was presented using a deterministic algorithm (Karlsson *et al.*, 2024).

Among meta-heuristic algorithms, ACOR is one of the representative methods for solving the optimization problem in cable routing domain (Socha & Dorigo, 2008). ACOR is a variant of ant colony optimization to solve continuous domain problems rather than discrete domain. In this study, RFACOR was utilized among several variations of ACOR (Zhou *et al.*, 2023), which showed improved performance compared to other variant algorithms.

2.2 Pathfinding algorithms

Pathfinding algorithms are utilized not only in routing but also in various domains such as gaming, robotics, and artificial intelligence (Nobes *et al.*, 2022). In 3D space, typical environments where pathfinding algorithms operate are octree and voxel grids. The octree has been extensively used in sectors, like the unmanned aerial vehicle domain, where analysis of vast areas is essential (Rodenberg *et al.*, 2016). Unlike the uniformly structured voxel grids, the octree is nonuniform, requiring separate algorithms to find neighboring nodes for pathfinding (Samet, 1989; Kang *et al.*, 1999; Nandari *et al.*, 2015).

The voxel grids environment, with its uniform structure in every direction, is relatively convenient for applying pathfinding algorithms. The A* algorithm is the most popular and classical algorithm that can be employed in voxel grids (Hart *et al.*, 1968). A* is a variant of the Dijkstra algorithm, where a heuristic cost considering the distance to destination is included during pathfinding. As a result, there are instances where weights tailored to the application domain are applied (Liu *et al.*, 2019). Theta*, one of the variations of A*, expands the node search area based on LOS and offers the advantage of returning a path closely resembling the actual optimal distance without direction constraints.

The JPS is a representative algorithm that effectively handles symmetric paths arising due to the uniform properties of voxel grids. JPS achieves a significant reduction in execution time with an efficient pruning strategy based on the symmetry of paths (Harabor & Grastien, 2011). There is a case that introduces a method of applying angle-based LOS judgment for path smoothing on the optimal route obtained through the JPS algorithm for mobile robot pathfinding (Luo *et al.*, 2022). In this study, we introduce a new algorithm that applies the LOS check process like Theta* to the search process of JPS.

Paths obtained from pathfinding algorithms in obstacle environments are combinations of straight paths, necessitating smoothing for a more fluid trajectory. Especially in robot path planning, studies have been proposed for path smoothing to ensure more natural movements. Previous research introduced algorithms that interpolate by inserting midpoints between path points. One study devised an algorithm that generates a B-spline path in environments with obstacles, and for the path segments that collide with obstacles, they inserted midpoints to avoid these collisions (Noreen, 2020). Another study proposed an interpolation method that smooths the B-spline by interpolating additional points when the maximum curvature is exceeded, building upon the conventional algorithm of adding midpoints for interpolation (Elbanhawi *et al.*, 2015). Similarly, this study uses a post smoothing algorithm to eliminate excessive path points for generating natural B-spline shape.

3. JPS–Theta*: LOS-based JPS algorithm

We introduce the JPS and Theta* algorithms in this section. Then we provide JPS–Theta*, a LOS-based JPS algorithm proposed in this paper, which is based on these two algorithms.

3.1 JPS algorithm

By performing pruning strategy in the expansion, JPS significantly decreases the search area in voxel grids and creates jump points upon identifying nodes with forced neighbors. The definition of the forced neighbor is as follows where the path is denoted as π, the parent node as p, arbitrary nodes x, y are introduced with parent node p, and any additional arbitrary node is denoted as n.

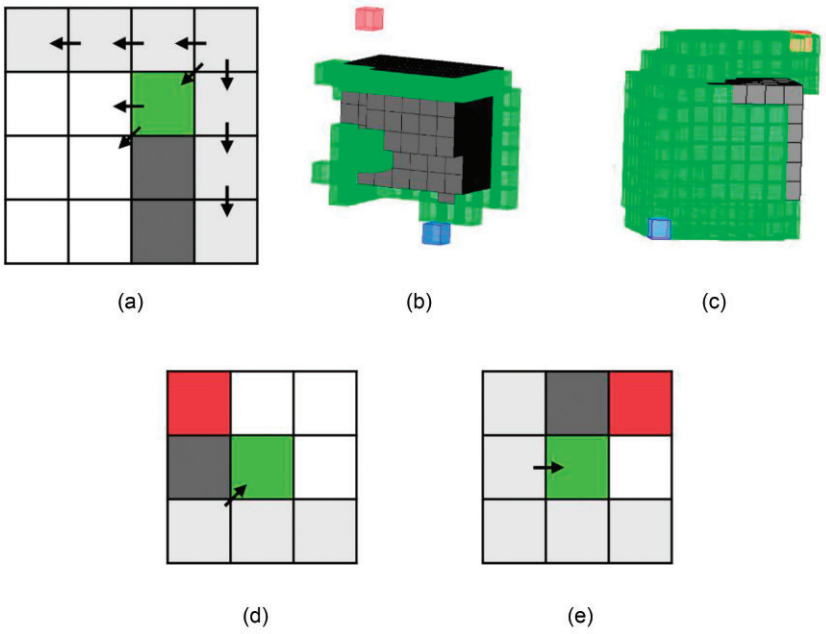

Figure 3: JPS algorithm overview and comparison: (a) pruned regions (gray) and a jump point (green), (b) and (c) compare the number of search nodes of JPS and A* within the same environment, (d) and (e) illustrate jump points recognized by a forced neighbor (red) during diagonal and straight moves, Obstacles are represented in dark gray.

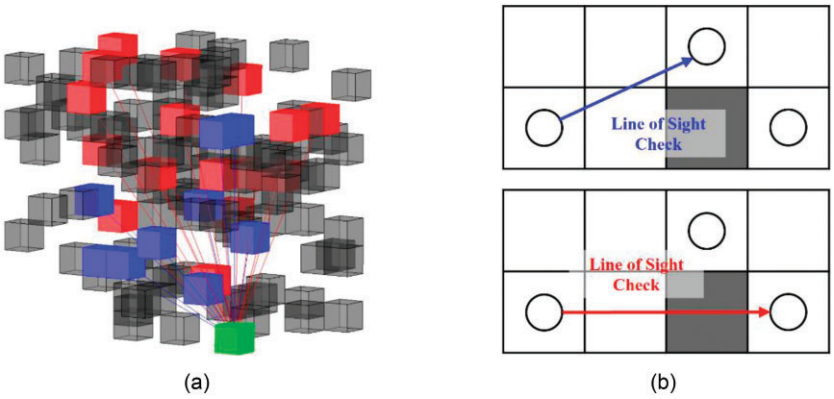

Figure 4: LOS check in 3D and 2D: (a) the red nodes represent nodes for which LOS has not been secured and the blue nodes represent nodes for which LOS has been confirmed, (b) illustrates LOS checks in 2D.

Definition 1 For $n \in neighbors(x)$, n is a forced neighbor if the following conditions hold (Harabor & Grastien, 2011).

(1) n is not a natural neighbor.
(2) When compared to the path $\pi = \langle p, x, n \rangle$ containing x, if the path $\pi' = \langle p, x, n \rangle$ is not valid due to obstacles and is shorter or equal, then n is considered a forced neighbor.

JPS performs pathfinding using only generated jump points when expansion, considering the distance between jump points and the destination as a heuristic cost. These characteristics are shown in Fig. 3. Jump points generated by forced neighbor are shown in Fig. 3a, whereas search nodes for JPS and A* are shown in Fig. 3b and c, respectively, within the same environment. When comparing the number of search nodes in Fig. 3b and c, it can be seen that JPS shows a lot less nodes than A*. Given its characteristics, JPS performs a faster search time than A* (Nobes *et al.*, 2022). Figure 3d and e show example of the forced neighbors found during the search in 2D space. The details of JPS, which include the pruning rule, detailed definition of forced neighbors, and jumping rule in 3D, can be found in the previous paper (Nobes *et al.*, 2022).

3.2 Theta* algorithm

Theta* is a variant of the A* algorithm. In A*, the optimal path is determined exclusively in fixed directions: 8 for 2D and 26 for 3D. Thus, path derived by A* may deviate from the true shortest path. To address this, Theta* combines the node expansion method of A* with LOS check. LOS check refers to verifying the LOS between two nodes, as shown in Fig. 4. The Bresenham algorithm, a well-known line-drawing algorithm, is employed for the LOS check.

Theta* check LOS between the child and parent of a reference node during node expansion, if LOS is confirmed, update the path between the parent node and the extended node, excluding intermediate nodes. As a result, it can find any-angle paths in its expansion and returns shorter paths than JPS and A*. The process of continuously removing intermediate nodes through LOS checks to simplify the path is referred to as PS (Post Smoothing).

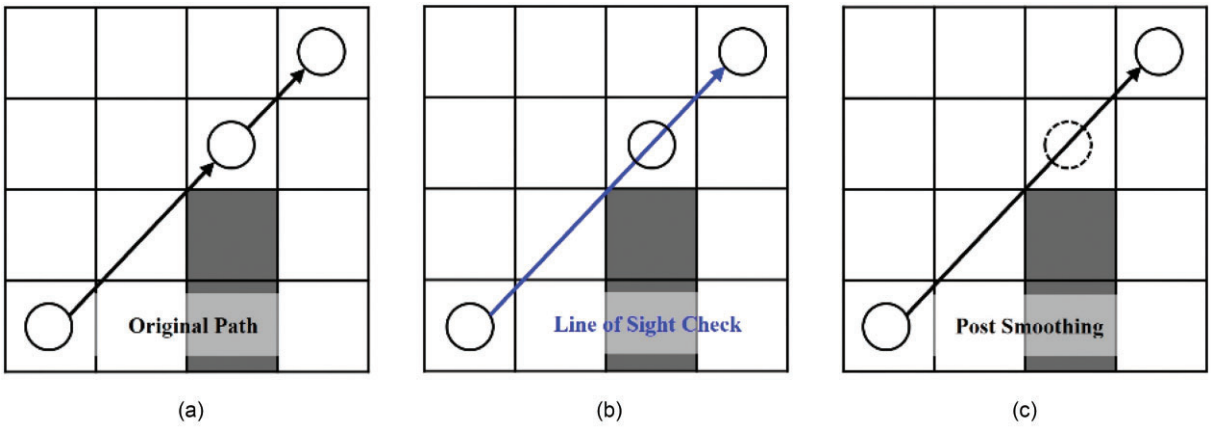

Figure 5: Post smoothing process based on LOS check.

PS can also be employed to remove intermediate nodes on the final path of JPS and A*. Applying PS to the final path obtained from A* and JPS can obtain any-angle path. Theta*, on the other hand, is known to provide shorter paths because it continuously evaluates the simplified path against other possible paths during node expansion and removes intermediate nodes. PS process is shown in Fig. 5.

3.3 Proposal of an improved JPS algorithm based on LOS

In this study, we aim to harness the strengths of both JPS and Theta* algorithms by combining them. In previous studies, various methods have been proposed to enhance JPS, but none have combined JPS with Theta* during its operation. Both algorithms are relatively recently proposed, and this combination is the first in our study. The expected benefits of this combination are as follows. First, we can expect faster search time based on the pruning strategy of JPS. Next, you can obtain any-angle path, shorter path with Theta* through LOS check.

We combined JPS and Theta* algorithms by adding LOS check during jump point generation in the node expansion of JPS. While running the JPS algorithm, intermediate nodes are deleted through LOS check to simplify the path. This approach is similar to applying PS to the final path obtained using JPS algorithm (JPS PS). However, the proposed approach fundamentally differs from the JPS PS in that it smooths the path during the algorithm not at the end, thus results are different.

Figure 6 shows the flowchart illustrating the sequence of jump point generation in the JPS–Theta* algorithm. This process connects the *Jump* process of JPS and the *UpdateVertex* process of Theta*. *Jump* is an important operation for identifying candidate nodes (i.e., jump point nodes) on the path. Through the *Jump* operation, JPS can effectively reduce the number of candidate nodes, which makes it a very efficient path search algorithm. Among the candidate nodes, the path is simplified as much as possible using the *UpdateVertex* based on LOS and these nodes are stored to continue the search. This simplified path is advantageous for generating a smoother and more natural cable path.

In this study, through the process of identifying jump points and updating the path, the LOS between the child nodes and parent nodes of the reference node is continuously checked to compare the simplified path with the original path and remove unnecessary intermediate nodes. The *UpdateVertex* method of Theta* is known to have a faster execution time than the postprocessing method of path simplification and to frequently return shorter paths (Daniel *et al.*, 2010). Figure 7 shows an example of identifying jump points and updating the path based on LOS.

The JPS–Theta* is a heuristic-based algorithm, so it has better generality than optimization-based algorithms, but it cannot always guarantee the optimal solution. Therefore, to achieve a specific goal (e.g., collision avoidance), optimization needs to be performed afterward. Nevertheless, the proposed JPS–Theta* has advantages in terms of time efficiency and path smoothness in general compared to optimization-based (e.g., GA-based) routing algorithms.

3.4 Comparison of pathfinding algorithms

To compare the performance of the existing pathfinding algorithms, such as JPS and Theta*, with the newly introduced JPS–Theta* algorithm in this study, it is essential to design appropriate experiments. In this section, we design experiments for comparing pathfinding algorithms. The programming language used was Python, and the hardware employed for the study consisted of an Intel(R) Core(TM) i7-12700K CPU and 32GB of RAM. We implement each algorithm and iteratively execute them. Subsequently, we compare and analyze the performance of each algorithm. In all experiments, at least one valid path was included to exclude scenarios where there was no path. The design space consisted of a voxel grid of a cubic environment with minimum and maximum points from (−200, −200, −200) to (200, 200, 200). The start and end nodes were set to correspond to the minimum and maximum points, respectively. Obstacle generation was conducted by controlling the number and maximum and minimum sizes of obstacles to have a density of 20 to 25% (Daniel *et al.*, 2010).

In this experiment, we compared JPS PS, Theta*, and JPS–Theta* algorithms. The results include path length, execution time, total angle of the path, and the change of direction. Here, the term "the change of direction" refers to the number of direction changes equal to or greater than 45 deg. Lastly, Theta* algorithm was implemented using a closed set. It is known that when a closed set is employed, it returns relatively longer distances but offers the advantage of reduced search time (Daniel *et al.*, 2010). In the 3D voxel grids, the reduction in search time becomes even more apparent due to the larger number of nodes to search compared to the 2D grids.

Figure 8 represents the experimental results. To simultaneously observe the average and variance of the experimental outcomes, we utilized box plots. In terms of search time, algorithms

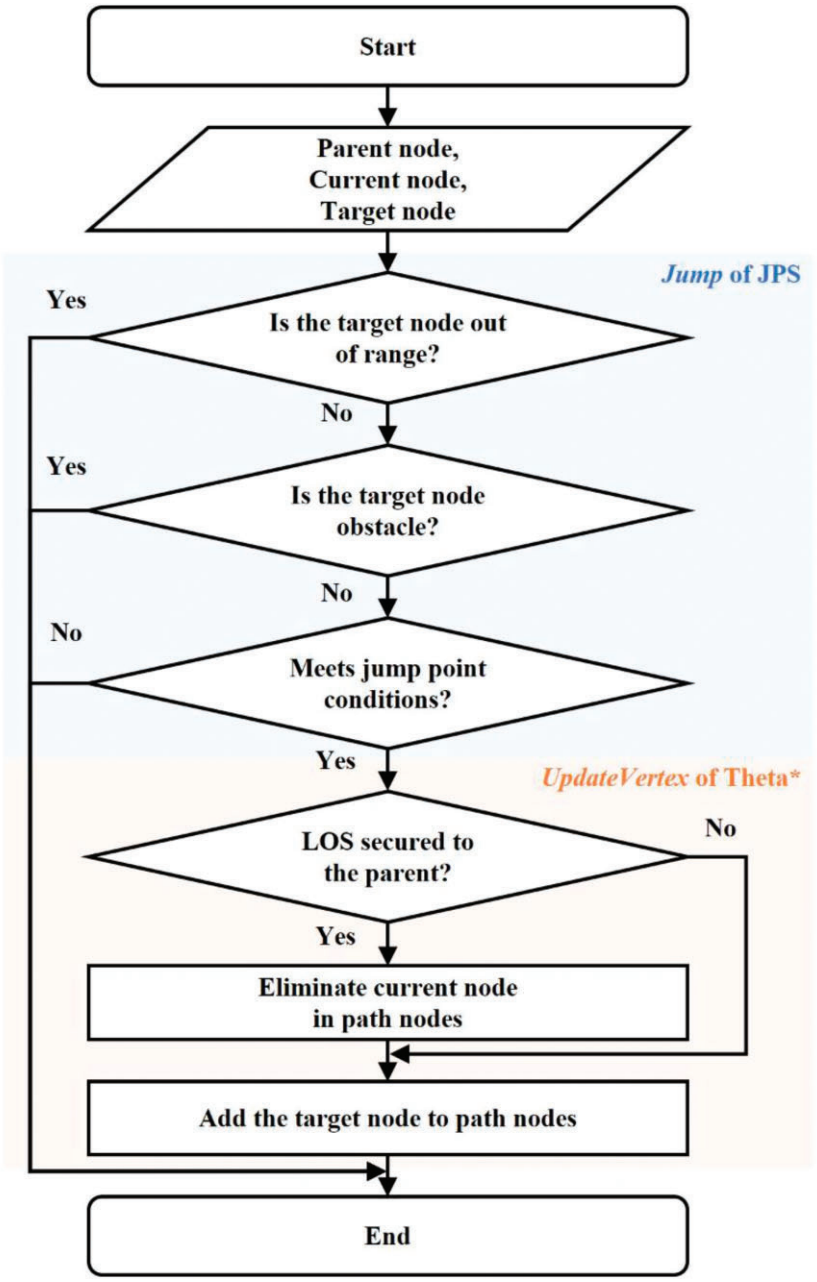

Figure 6: Flowchart of the proposed JPS–Theta*.

based on JPS exhibit relatively lower search times compared to Theta*. Particularly, as the grid resolution increases, a significant difference in search time becomes evident. In a voxel grid environment of 100*100*100, JPS–Theta* was about 18 times faster than Theta* on average, and JPS–Theta* was about two times faster than JPS PS. This result shows that Theta*'s *UpdateVertex* method is effective in reducing execution time compared to the PS method. This result is a trend that also appears in the existing Theta* literature (Daniel *et al.*, 2010). This reflects the characteristic of JPS, which considers only jump points as path points. From a path distance perspective, there is no significant difference among the three algorithms. However, the Theta*-related algorithms tend to yield slightly shorter paths. However, when examining the change of direction changes, it was evident that both Theta* and JPS–Theta* have lower values compared to JPS PS. Excessive directional changes in a path can strain the cable, so they should be avoided. Therefore, Theta* and JPS–Theta*, which have relatively fewer direction changes, can be considered suitable for automatic cable routing. Therefore, it can be seen that JPS–Theta*, which has appropriate changes in direction and stable changes in angle, is also advantageous for future B-spline optimization.

According to the experimental results, our proposed JPS–Theta* algorithm exhibits similarity with JPS and Theta* regarding distance and angle. Moreover, it combines the advantages of both algorithms in terms of search time and directional changes, offering a logical alternative. Especially considering its minimal time consumption and reduced abrupt direction changes, our proposed method appears to be a suitable pathfinding algorithm for automatic cable routing. Nevertheless, it is important to note that

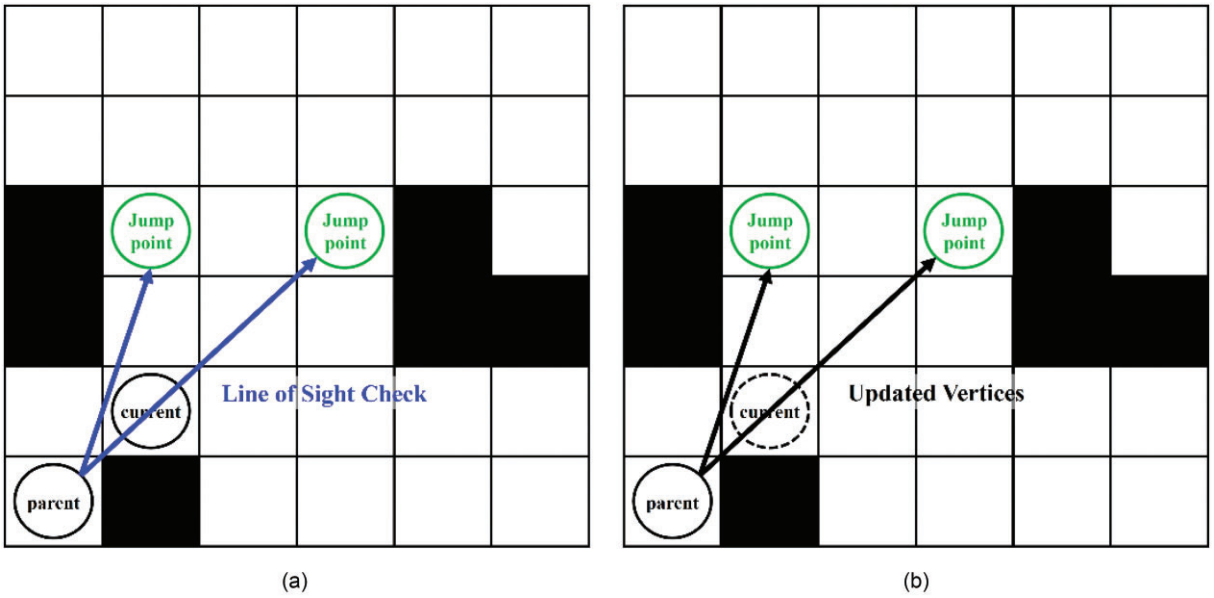

Figure 7: Example of creating a jump point in JPS–Theta*.

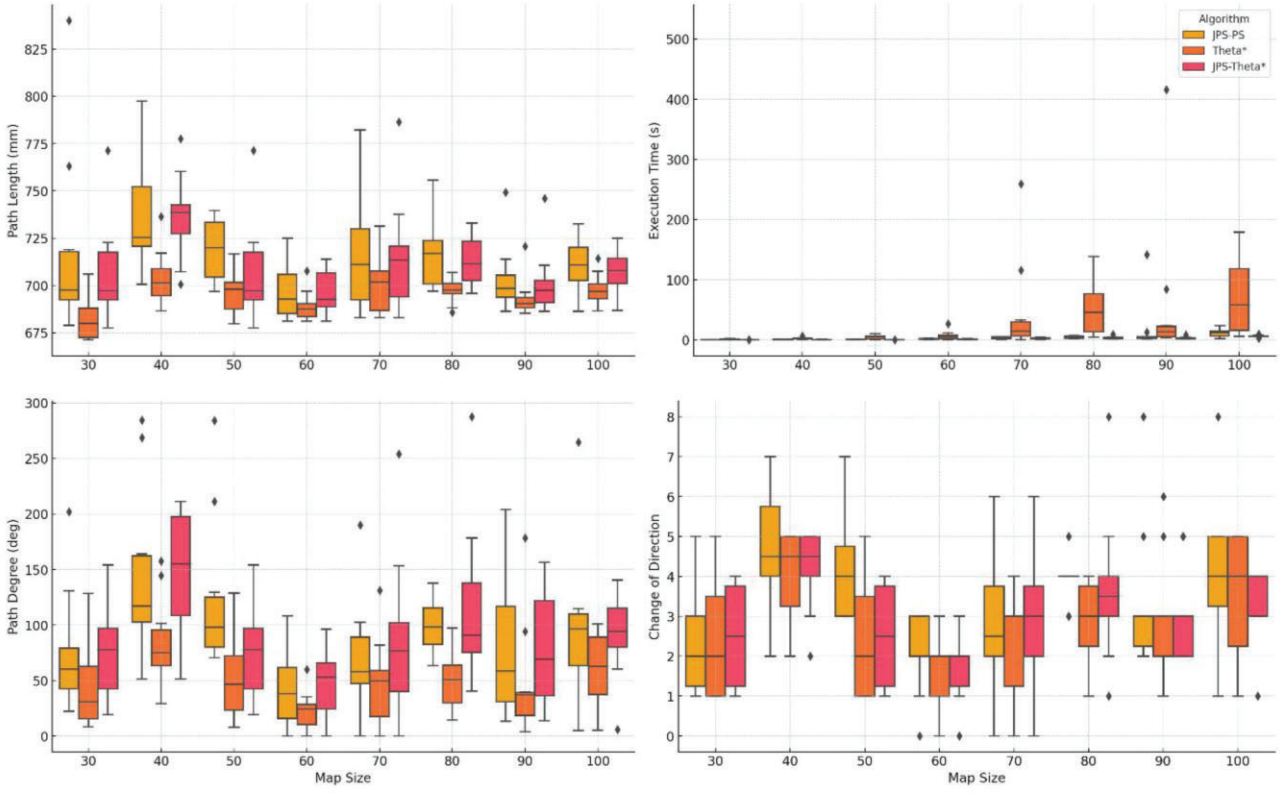

Figure 8: Comparison of time, length, change of direction, and angle between JPS PS, Theta*, and JPS–Theta*.

the results of pathfinding algorithms can vary depending on the operational environment. Therefore, additional experiments and evaluations across various environments are necessary.

Environments where jump points can be easily generated can cause inefficiency of JPS-related algorithms. Of course, cases may vary depending on the degree of optimization of forced neighbor determination or *Jump* operations. Figure 9 shows that the results we implemented have different tendencies depending on the environment. Figure 9a is a boxplot graph of the execution time in each environment. The left side is the result from the current experiment (clustered), and the right side is the graph from the comparison environment (scattered). Figure 9b is the visualization result of each environment. In the comparison environment where jump points can be easily generated, we can see that the

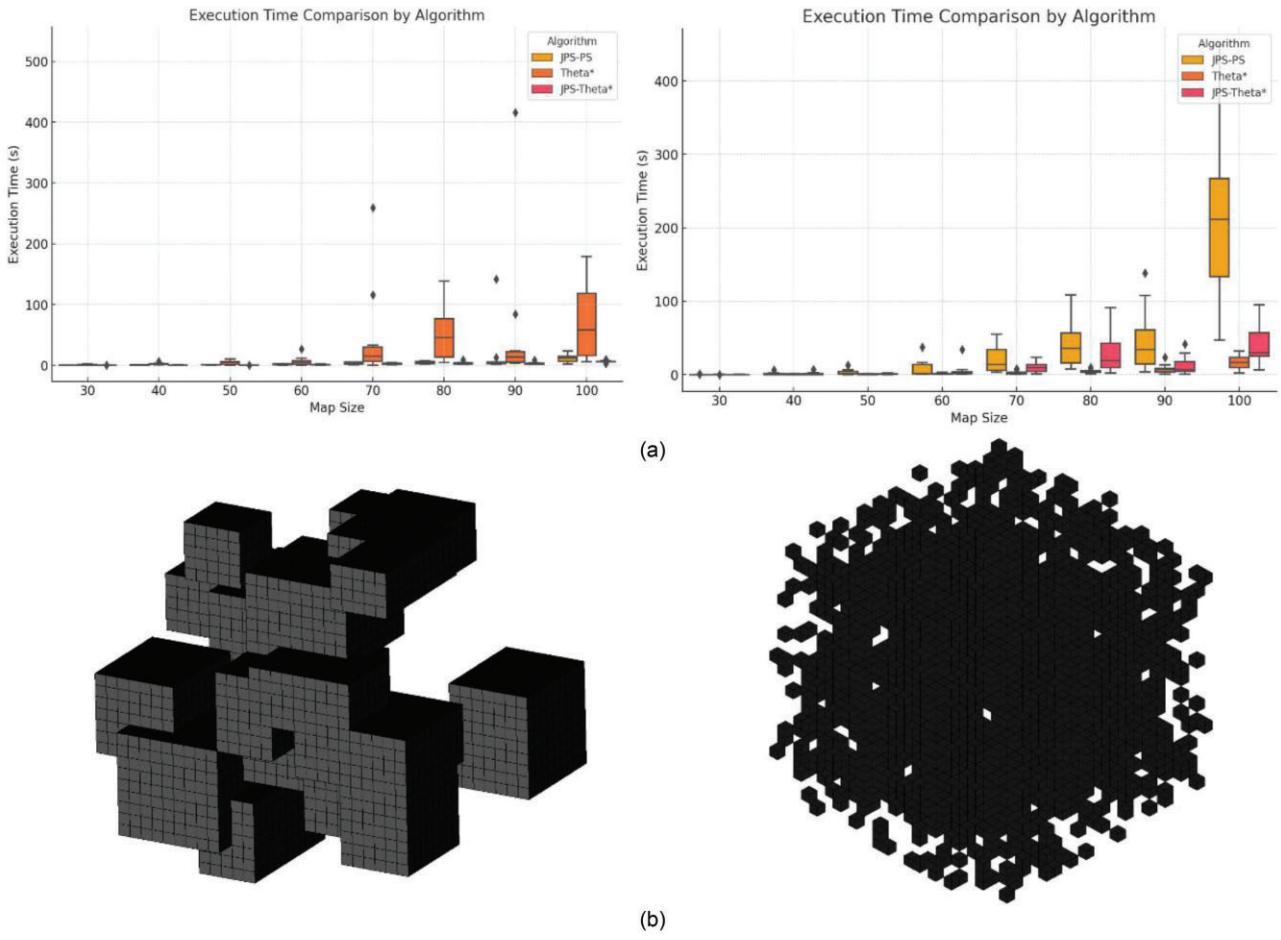

Figure 9: Comparison of execution times for different obstacle environments.

execution time of the JPS-related algorithm is slower than the results from the comparison environment.

4. B-spline optimization for collision avoidance

4.1 Problem definition

In this section, we propose an optimization method to address potential collision issues that may arise during the generation of a B-spline based on the initial path points obtained from pathfinding algorithms. When obtaining the initial cable, the resulting path points of the pathfinding algorithm are used as inputs to the B-spline interpolation function. By utilizing the simplified path through JPS–Theta* during B-spline interpolation, a more natural initial cable path can be obtained. To solve the optimization problem, this study applies the metaheuristic algorithm known as RFACOR (Zhou *et al.*, 2023). RFACOR is a modified version of the traditional ACOR algorithm, designed to solve continuous domain problems. It exhibits improved convergence patterns and mitigates the issue of easily falling into local minima (Zhou *et al.*, 2023). RFACOR provides an appropriate solution for cable optimization problems that must satisfy complex design constraints.

Figure 10 illustrates the potential issues that may arise when generating a B-spline based on initial path points. When using initial path points for B-spline interpolation, collisions with obstacles can occur during the conversion of straight paths into curves, and it cannot guarantee the satisfaction of design constraints. To address this, this study proposes a cable shape optimization methodology that comprehensively considers cable collisions, minimum curvature radius, and length.

4.2 Method of adding new path points for collision avoidance

To quickly find a solution that minimizes obstacle interference, an appropriate range of motion is required. In this study, we propose a method to expand the range of motion of the cable shape by adding new path points based on collision points. Here, collision points refer to points included in obstacle voxels among the point clouds sampled on the surface of the cable shape. Figure 11 visually illustrates the procedure for adding new path points to the cable based on collision points, and Fig. 12 shows the corresponding flowchart.

First, collision points are obtained based on the point clouds sampled from the cable surface. Next, to divide the collision points, clustering is performed on these points. In this study, the DBSCAN algorithm was utilized for clustering. The DBSCAN parameters were set as follows: the neighbor radius ε_{min} was set to 10, and the minimum number of neighbors n_{nei} was set to 2. Finally, the center point of the cluster becomes a new path point and is inserted into the existing path. If the centroid of each cluster is within a certain radius of an existing path point, it is excluded from being added as a new path point. The radius r_{min} was set to 20. This exclusion criterion is applied because adding path points too close to existing ones does not help expand

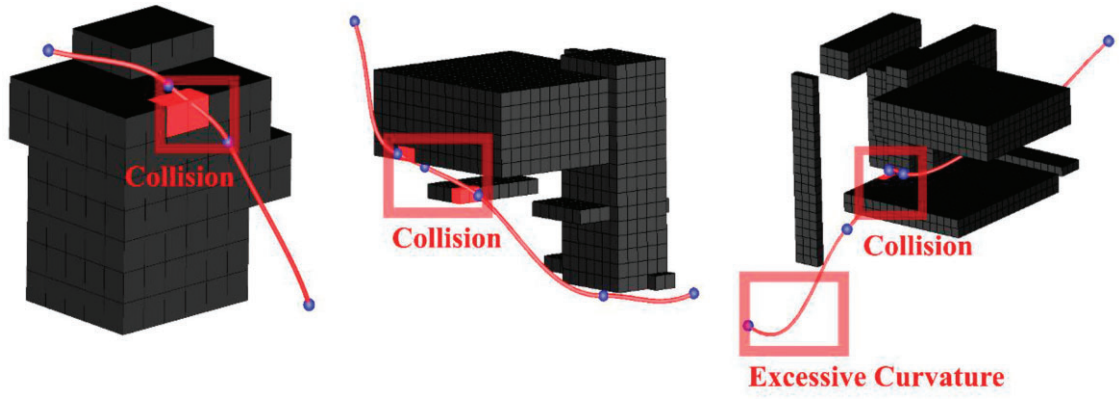

Figure 10: Examples of B-spline generation using initial path points.

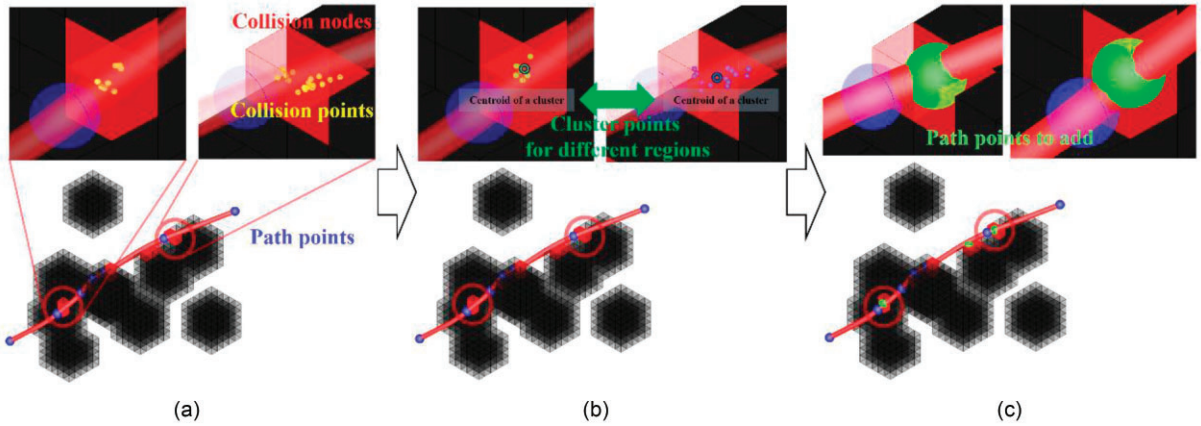

Figure 11: Visual explanation of process of adding new path points.

the range of motion of the cable shape during the optimization process.

4.3 Problem formulation

To comprehensively consider cable collisions, minimum curvature radius, and length, we propose cost functions as shown in equation (1). The cost function C_{length} takes into account the cost-effectiveness of the cable. Generally, as the cable length increases, so do the material and installation costs; therefore, considering the cable length in the optimization process is crucial. C^{min}_{radius} is a cost function that considers the stability of the cable. The minimum curvature radius is a factor considered for the physical stability and reliability of the cable. An excessively small minimum curvature radius imposes excessive stress on the cable structurally, increasing the likelihood of damage. Curvature is defined as the reciprocal of the curvature radius. Finally, $C_{collision}$ is a cost function that considers the physical collisions of the routed cable model. To achieve a feasible routing path, it is necessary to minimize the physical collisions of the cable. $C_{collision}$ samples the point clouds P on the surface of cables and uses the *CollisionCheck*() function to determine the number of points included in the obstacle voxels, utilizing this count as the value for $C_{collision}$. The *CollisionCheck*() function identifies a point as a collision point if the input point is included in the obstacle voxel.

The solution *s* refers to the displacement vector and tangent vector adjustment values for the intermediate points, excluding the start and end points, of the initial path points after pathfinding. $s_{initial}$ denotes the solution corresponding to the B-spline generated based on the initial path. During the optimization process, the positions and slopes of the path points are adjusted for optimization. The objective function is primarily composed of $C_{collision}(s)$ to minimize cable interference. To minimize obstacle interference while preserving the initial cable length and minimum curvature radius, the constraint violations $g_{length}(s)$, $g^{min}_{radius}(s)$ are defined as shown in equation (2). The values on the right-hand side of the constraint violation equations are determined experimentally and can be adjusted according to the requirements. The penalty function is constructed in a basic static penalty form. R_{length}, R^{min}_{radius} in equation (3) are the penalty coefficients related to the length and minimum curvature radius, respectively. Finally, the solution is evaluated and optimized according to equation (4).

$$C_{length}(s) = L, \quad C^{min}_{radius}(s) = \max_{W \in w_i} \{\kappa(w_i)\},$$
$$C_{collision}(s) = CollisionCheck(P) \tag{1}$$

$$\begin{aligned} &\text{Minimize} \quad f(s) = C_{collision}(s) \\ &\text{Subject to} \quad g_{length}(s) = \frac{C_{length}(s)}{C_{length}(s_{initial})} \le 1.5 \\ &\qquad\qquad g^{min}_{radius}(s) = \frac{C^{min}_{radius}(s)}{C^{min}_{radius}(s_{initial})} \le 2.0 \end{aligned} \tag{2}$$

$$R_{length} = C_{length}(s_{initial}), \; R^{min}_{radius} = C^{min}_{radius}(s_{initial}) \tag{3}$$

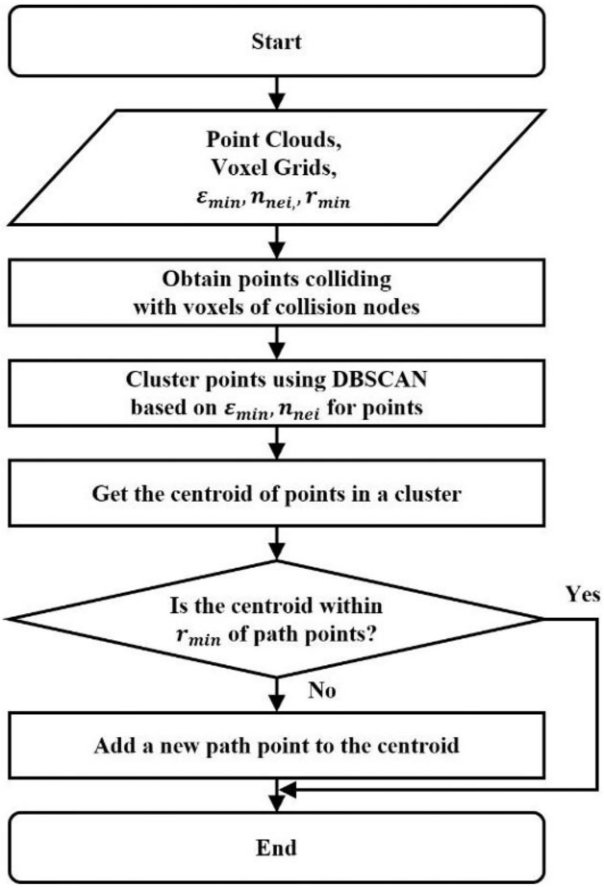

Figure 12: Flowchart of how to add path points based on collision points.

$$F(s) = f(s) + \sum_j R_j g_j(s) \tag{4}$$

4.4 Experiment on B-spline optimization algorithm for collision avoidance

In this section, experiments are conducted to verify the effectiveness of the proposed cable optimization method. In these experiments, optimization is performed based on the initial paths obtained after running JPS–Theta* 10 times in random environments ranging from 10 * 10 * 10 to 50 * 50 *50. The results are compared in terms of cable length increase rate, maximum curvature increase rate, and collision removal rate. The OpenCascade library was used as a modeling kernel to construct and visualize the 3D path exploration environment and cable geometry. Python was used as the programming language, and the hardware used in the study consisted of an Intel(R) Core(TM) i7-12700K CPU and 32GB of RAM. The obstacle environment differs from that of the pathfinding environment. The boundary of the random environment is a cube with a side length of 400, and the generated obstacles consist of rectangular prisms composed of 64 to 1728 obstacle nodes. Their positions are determined randomly, with user-specified distance constraints centered on the midpoint of the box. the parameters for RFACOR were selected as shown in Table 1 (Zhou *et al.*, 2023). fes_{max} refers to the maximum number of function evaluations, and ub, lb denote the upper and lower bounds of the solution range, respectively. k represents the size of the solution archive used in the ACOR algorithm. The solution archive stores sampled solutions during the search process of ACOR. The dimension of the solution is six times the number of path points to adjust the displacement and slope of the path points. n represents the size of the dimension of the solution. m denotes the number of ants, which determines the number of sampling iterations before the solution archive is updated. Once the required number of sampling iterations is reached, solutions are sorted based on their objective function values, and those with high objective function values, except for the top k solutions, are discarded to improve the solutions. q determines the distribution during the sampling process; as q increases, the distribution used for sampling becomes more uniform. ξ generally has a range $\xi > 0$. A higher value of ξ leads to quicker forgetting of poor solutions during the search process (Socha & Dorigo, 2008). The maximum number of function evaluations fes_{max} is set to 1500 to allow sufficient exploration. ub and lb are set to 40 and −40, respectively, to limit the exploration of too wide a range. Solutions with too wide a range are unrealistic and do not satisfy the design conditions. If the archive size k is not sufficient compared to the solution dimension size, the results may not converge during the optimization process. To ensure sufficient archive size, 35 was added to the solution dimension size and used as the solution archive size. For stable convergence, the remaining values in Table 1 were used as references in the existing ACOR study (Socha & Dorigo, 2008).

Table 1: RFACOR and ACOR parameter settings.

Parameter	Value
fes_{max}	1 500
ub	40
lb	−40
k	$n + 35$
m	10
q	0.7
ξ	0.8

Table 2: Performance comparison results for comparing the performance of ACOR and RFACOR.

Parameter	RFACOR	ACOR
Best Fitness Value	0.0124	20.335
Average Fitness Value	8.0651	25.062
Average Convergence Generation	5.4	5.8

In this study, we propose the application of RFACOR, which is an improvement over the existing ACOR, to perform cable optimization in complex environments. Compared to ACOR, RFACOR shows better optimization performance by solving the problem of converging to local optimization. Table 2 compares the average values of Best Fitness Value, Average Fitness Value, and Average Convergence Generation after 10 trials in a 50 * 50 * 50 environment for comparing ACOR and RFACOR in the cable optimization problem (Liu *et al.*, 2024). Best Fitness Value means the most appropriate objective function value in the optimization process. Average Fitness Value is the average of the function values (fitness values) recorded by each algorithm while performing optimization and represents the average performance during the entire iteration. Average Convergence Generation compares the number of generations when the algorithm converges, that is, the fitness value no longer changes. Best Fitness Value and Average Fitness Value mean that the lower the value, the better the

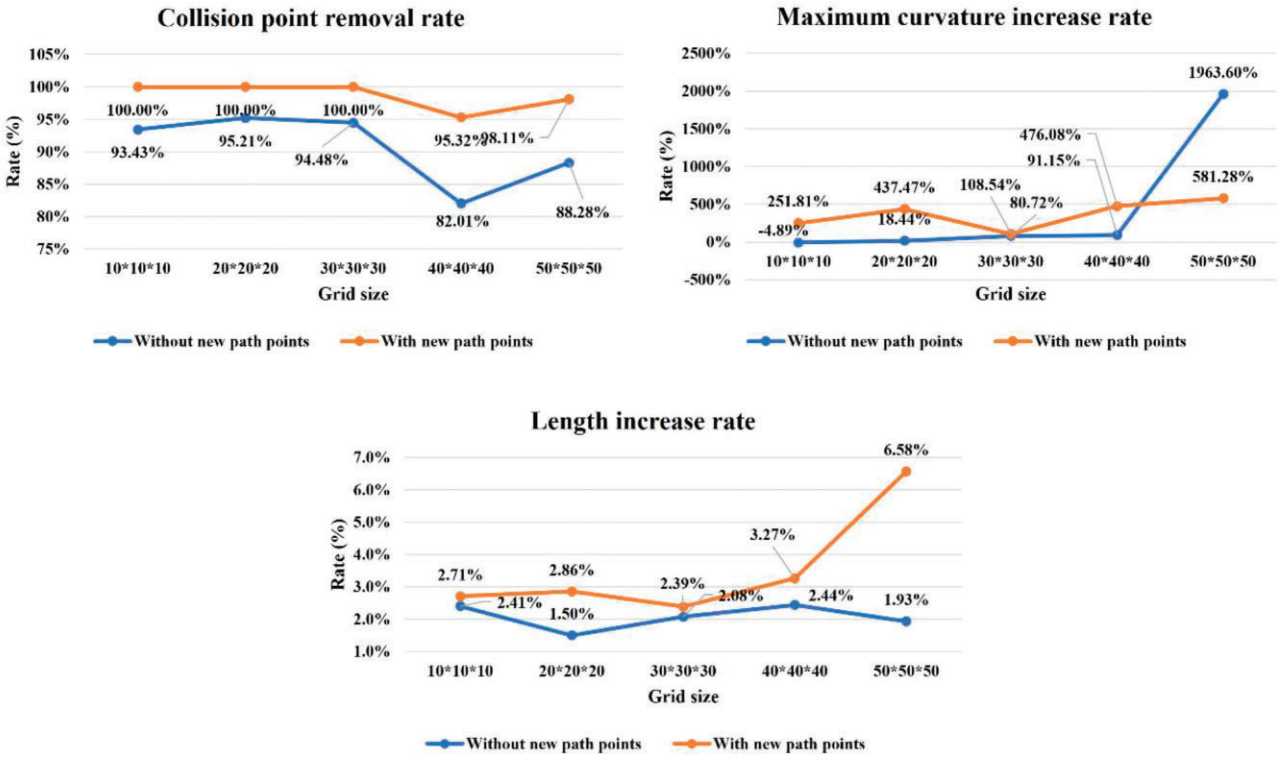

Figure 13: Experimental results of cable optimization with and without new path points.

performance. This means that RFACOR can search for a better optimal value, and it can be seen that it searches for an appropriate value on average. On the other hand, Average Convergence Generation shows a larger ACOR value, which is judged to be the result of converging too quickly during the search process (convergence to the local optimum).

Figure 13 numerically compares the optimization results with and without new path points. The experimental results clearly show that adding path points significantly increases the collision removal rate, i.e., less collision. This indicates that adding path points is advantageous in expanding the range of motion during optimization, thereby removing obstacle interference. Although the length increase rate was generally higher when path points were added, the difference was not substantial, and in both cases, there was not much difference from the original cable length. Except for the last case, we can see that the cases where path points are added have a larger maximum curvature. This suggests that in complex random environments, the optimization prioritized obstacle avoidance, resulting in an increase in maximum curvature. Despite the increase in maximum curvature, all cases remained within design constraints. In conclusion, the proposed optimization method effectively minimized obstacle interference with a high success rate in complex random environments, and the minimal difference in length compared to the initial cable path demonstrates its efficacy.

To qualitatively verify the experimental results, the optimization outcomes in random obstacle environments were visualized. Figure 14 shows some examples of optimization experiments conducted with varying voxel resolutions. In all cases, obstacle interference was eliminated through optimization. Figure 15 presents some examples of optimization experiments conducted with different obstacle ratios. In these cases, as well, obstacle interference was eliminated in all instances. However, in some cases, although obstacle interference was removed, the cable length increased unnecessarily. Nevertheless, the length increase rate did not exceed 50% in any case. These examples demonstrate that the proposed method robustly eliminates collisions in various environments.

5. Case study of an electrical panel

In this section, we present a case study applying the proposed method to a real-world scenario. The selected case is an electrical panel design model. Cable routing in electrical panels is a common example of cable routing. We utilized the OpenCascade library as a modeling kernel for the construction and visualization of a 3D pathfinding environment as well as cable shapes. We used Python as a programming language. The hardware employed for this research consists of Intel(R) Core(TM) i7-12700K CPU and 32GB of RAM. If the path is not generated because the routable space is insufficient, B-spline optimization cannot be performed. However, this is a problem of the design environment rather than the proposed method, so if the path finding fails, the design environment will need to be changed. In an actual design environment, users can specify additional intermediate waypoints. Figure 16 shows the results of applying the proposed method to the panel design case. Figure 16a illustrates the B-spline cable shape generated based on the initial path points by the proposed pathfinding algorithm. The red lines represent cables with collisions, while the green lines represent cables without collisions. Figure 16b shows the results of optimizing the cables with collisions. The optimization results successfully resolved the obstacle interferences.

Figure 17 explains how to generate a harness. In Figure 17a, a location to generate a harness is selected when routing is already completed. Then, a routable node is identified based on

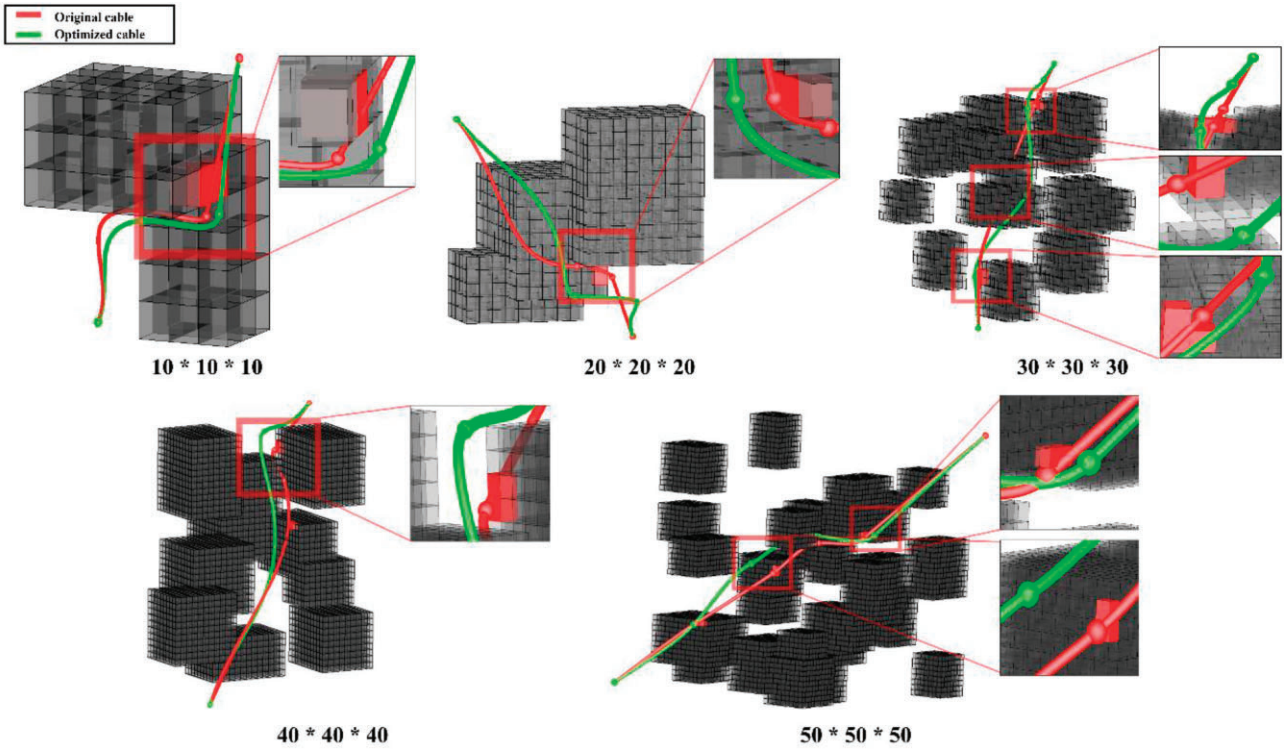

Figure 14: Examples of comparisons between original and optimized cables as voxel resolution increases.

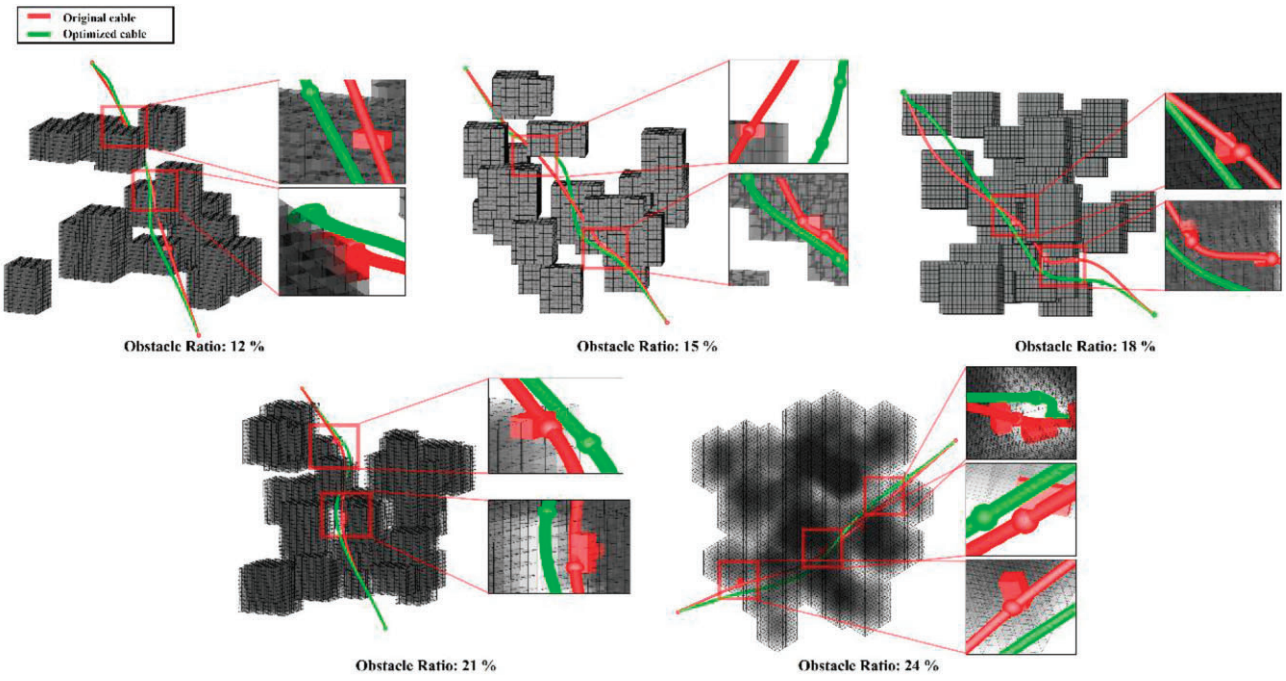

Figure 15: Examples of comparisons between original and optimized cables as obstacle ratio increases.

the input/output position and direction information of the harness, and a harness and internal cables are generated by performing pathfinding based on the node in Fig. 17b. Next, in Fig. 17c, pathfinding is performed on the intermediate path to connect the cable input/output terminals and the internal cables, and the resulting paths are merged. Figure 17d shows each result for different harness input/output information.

Finally, the results of performing cable routing, including harnesses, using the proposed method are shown in Fig. 18. Harnesses are used to bundle cables with similar paths to ensure space efficiency. In this process, cables and harnesses must exist as separate entities, and the cables included in a harness must align in their paths within the harness. Therefore, the start and end points of the harness were selected as waypoints for

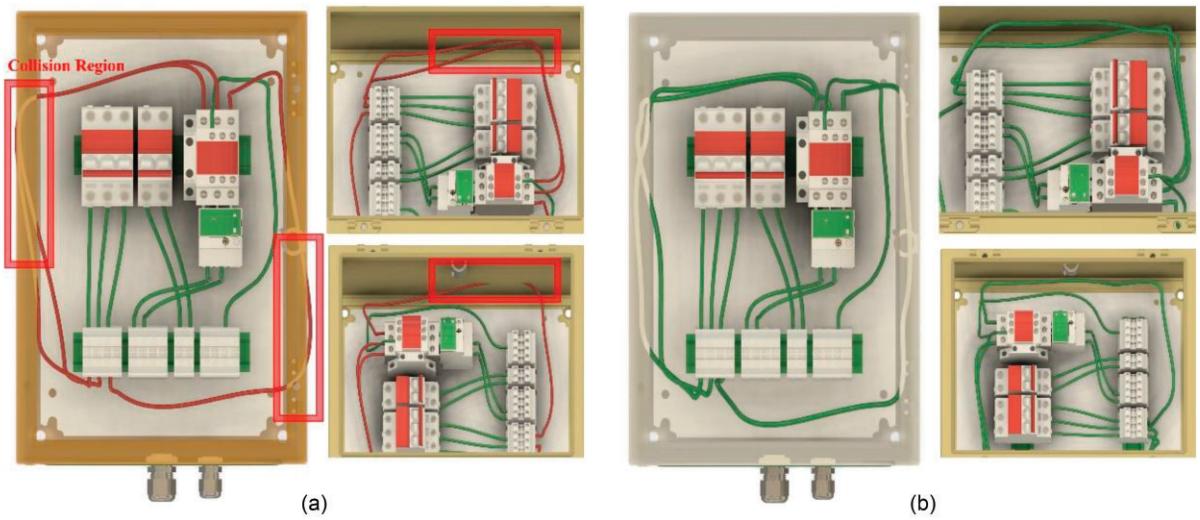

Figure 16: Results of applying the proposed method to electrical panels: (a) before optimization, (b) after optimization.

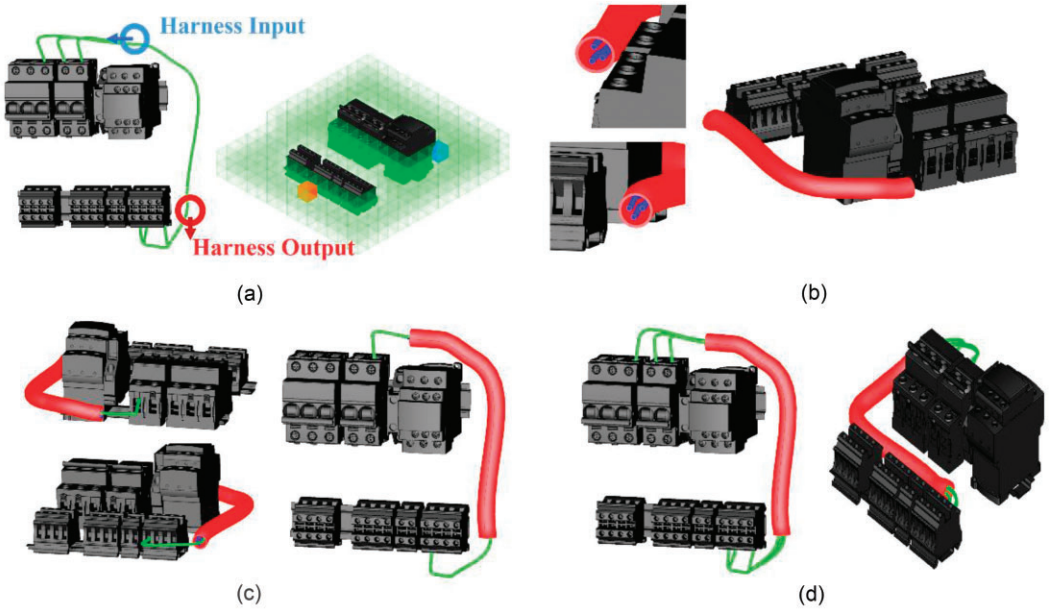

Figure 17: Routing process including harness creation.

the pathfinding process. Optimization was performed in cases of collisions, and ultimately, a successful routing design was achieved in the electrical panel design environment. The proposed automatic cable routing method is effective in identifying and eliminating obstacle interferences, making it particularly suitable for complex and confined environments like panels. Consequently, it is expected that the proposed method can reduce time wastage due to repetitive tasks in the cable routing design process.

In the case study, the total execution time of the routing process including the harness is 23 s. To provide an appropriate comparison, the execution time was compared by manually routing the cables. The manual routing was performed using Autodesk Fusion. The experiment was conducted after learning and practicing the routing method in advance. When routing a single cable manually, the execution time was 1 min and 41 s. Considering that the case study requires routing >10 cables and harnesses, we can see a huge advantage in the execution time through automatic routing.

6. Conclusions

Cable routing is a crucial aspect of electrical wiring design across various industries, requiring significant time and effort. The increasing demand for electrical and electronic systems, such as electric vehicles, has heightened the need for automatic cable routing. However, the use of automatic cable routing in 3D design remains limited.

In this study, we propose a new pathfinding algorithm, JPS–Theta*, which combines the traditional JPS and Theta* algorithms to reduce routing time and improve the final path. JPS–Theta* is advantageous in generating natural cables by eliminating unnecessary path points. Experiments demonstrated that JPS–Theta* reduces execution time compared to Theta* and improves the

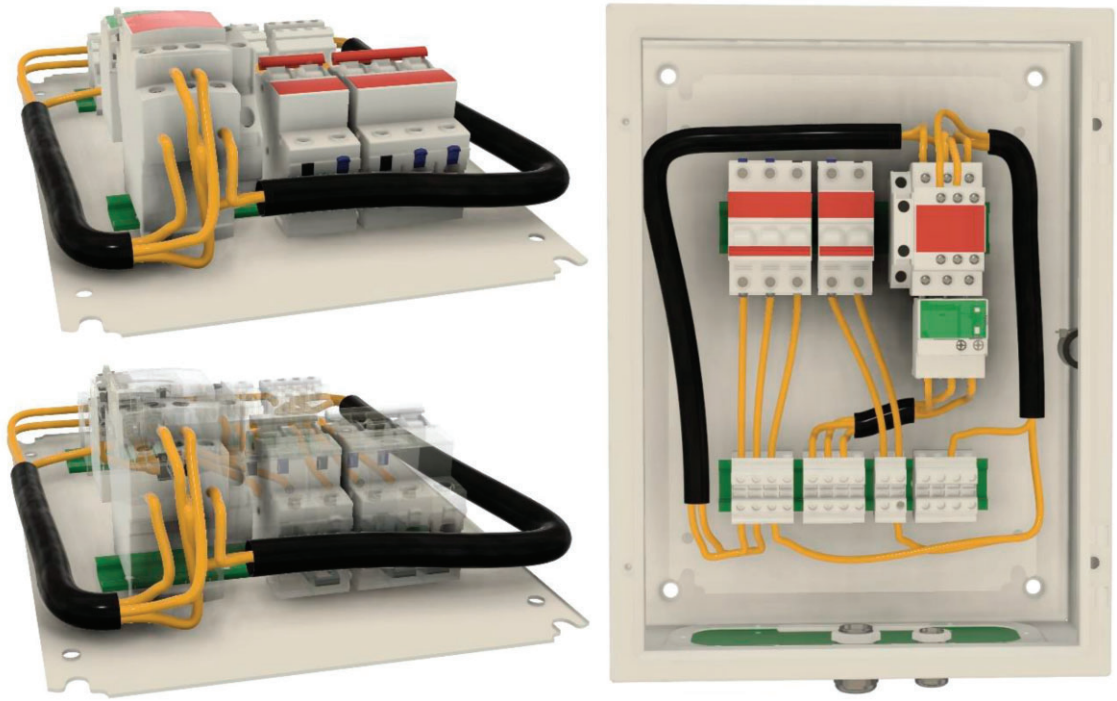

Figure 18: Cable routing example including harness using the proposed method.

length and angle of the final path compared to JPS PS. These results indicate that JPS–Theta* can be an appropriate compromise between Theta* and JPS.

Next, we proposed a B-spline optimization algorithm aimed at obtaining a cable shape that avoids collisions while meeting design constraints. The proposed algorithm performs optimization on the initial cable shape generated from the initial path obtained through the pathfinding algorithm, adjusting path points to ensure the cable conforms to design conditions. Cable optimization experiments in random environments quantitatively and qualitatively confirmed that the algorithm effectively eliminates obstacle interference in complex environments. Even if some collisions remain in the optimization results, modifying the cable after the interference is somewhat removed is more efficient than modifying the initial cable. Although it is ideal to completely remove interference in a complex design environment, modifying the cable after minimizing the collisions within a reasonable amount of time can be an efficient approach to reduce the design time.

The main contributions of this study can be summarized as follows. First, we presented a new algorithm, JPS–Theta*, which improves upon existing pathfinding algorithms and demonstrated its superiority. Second, we proposed B-spline optimization to generate cable shapes that meet design constraints and showed the robustness of the proposed method through experiments. Finally, we applied and analyzed the proposed method in a real-world electrical panel design. The proposed method is expected to be effectively applied to the electrical and electronic 3D design process, providing essential guidelines for cable routing.

We propose the following future research opportunities. First, there is a need for performance improvement in the B-spline optimization process. Next, automatically finding the optimal voxel resolution is necessary. Additionally, we will address collisions among cables in very narrow spaces. An automated method for forming optimal cable bundles for multiple cables is required. Finally, we plan to apply deep reinforcement learning to the pathfinding and B-spline optimization processes.

Conflict of interest statement

All authors declare that they have no conflicts of interest.

Author contributions

Kunchan Kim (Conceptualization, Methodology, Investigation, Software, Validation, Formal analysis, Data Curation, Visualization, Writing—original draft), Yeongjun Yoon (Methodology, Software), Byung Chul Kim (Investigation, Software, Data Curation), Jongguk Kim (Investigation, Funding acquisition), Soonhung Han (Investigation, Writing—original draft, Writing—review & editing), and Soonjo Kwon (Conceptualization, Methodology, Investigation, Resources, Project administration, Supervision, Writing—original draft, Writing—review & editing)

Acknowledgments

This work is supported by the Korea Agency for Infrastructure Technology Advancement (KAIA) grant funded by the Ministry of Land, Infrastructure and Transport (grant RS-2021-KA163348).

References

Chan, A. L. S., Hanby, V. I., & Chow, T.-T. (2007). Optimization of distribution pipping network in district cooling system using genetic algorithm with local search. *Energy Conversion and Management*, **48**, 2622–2629. https://doi.org/10.1016/j.enconman.2007.05.008.

Daniel, K., Nash, A., Koenig, S., & Felner, A. (2010). Theta*: Any-angle path planning on grids. *Journal of Artificial Intelligence Research*, **39**, 533–579. https://doi.org/10.1613/jair.2994.

Dijkstra, E. W. (2022). A note on two problems in connection with graphs. In *Edsger Wybe Dijkstra: His Life, Work, and Legacy* (pp. 287–290). ACM.

Elbanhawi, M., Simic, M., & Jazar, R. N. (2015). Continuous path smoothing for car-like robots using B-spline curves. *Journal of Intelligent & Robotic System*, **80**, 23–56. https://doi.org/10.1007/s10846-014-0172-0.

Fan, X., Lin, Y., & Ji, Z. (2006). The ant colony optimization for ship pipe route design in 3D space. *2006 6th World Congress on Intelligent Control and Automation*. IEEE, 3103–3108. https://doi.org/10.1109/WCICA.2006.1712938.

Harabor, D., & Grastien, A. (2011). Online graph pruning for pathfinding on grid maps. In *Proceedings of the AAAI Conference on Artificial Intelligence* (pp. 1114–1119). AAAI Press.

Hart, P. E., Nilsson, N. J., & Raphael, B. (1968). A formal basis for the heuristic determination of minimum cost paths. *IEEE Transactions on System Science and Cybernetics*, **4**, 100–107. https://doi.org/10.1109/TSSC.1968.300136.

Hermansson, T., Bohlin, R., Carlson, J. S., & Soderberg, R. (2013). Automatic assembly path planning for wiring harness installations. *Journal of Manufacturing Systems*, **32**, 417–422. https://doi.org/10.1016/j.jmsy.2013.04.006.

Hermansson, T., Bohlin, R., Carlson, J. S., & Soderberg, R. (2016). Automatic routing of flexible 1D components with functional and manufacturing constraints. *Computer-Aided Design*, **79**, 27–35. https://doi.org/10.1016/j.cad.2016.05.018.

Kang, S.-S., Myung, S., & Han, S.-H. (1999). A design expert system for auto-routing of ship pipes. *Journal of Ship Production*, **15**, 1–9. https://doi.org/10.5957/jsp.1999.15.1.1.

Karlsson, T., Ablad, E., Hermansson, T., Carlson, J. S., & Tenfalt, G. (2024). Automatic cable harnesses layout routing in a customizable 3D environment. *Computer-Aided Design*, **169**, 103671. https://doi.org/10.1016/j.cad.2023.103671.

Kim, S., Choi, T., Kim, S., Kwon, T., & Lee, T. H. (2021a). Sequential graph-based routing algorithm for electrical harnesses, tubes, and hoses in a commercial vehicle. *Journal of Intelligent Manufacturing*, **32**, 917–933. https://doi.org/10.1007/s10845-020-01596-9.

Kim, S., Kim, S., Choi, T., Kwon, T., & Lee, T. H. (2021b). Automatic design system for generating routing layout of tubes, hoses, and cable harnesses in a commercial truck. *Journal of Computational Design and Engineering*, **8**, 1098–1114. https://doi.org/10.1093/jcde/qwab034.

Kim, Y. S., Lee, K. H., Nam, B. W., & Han, Y. S. (2023). Application of reinforcement learning based on curriculum learning for the pipe auto-routing of ships. *Journal of Computational Design and Engineering*, **10**, 318–328. https://doi.org/10.1093/jcde/qwad001.

Liu, Q., & Wang, C. (2015). A graph-based pipe routing algorithm in aero-engine rotational space. *Journal of Intelligent Manufacturing*, **26**, 1077–1083. https://doi.org/10.1007/s10845-013-0840-0.

Liu, C., Mao, Q., Chu, X., & Xie, S. (2019). An improved A-star algorithm considering water current, traffic separation and berthing for vessel path planning. *Applied Sciences*, **9**, 1057. https://doi.org/10.3390/app9061057.

Liu, C., Wu, L., Li, G., Xiao, W., Tan, L., Xu, D., & Guo, J. (2024). AI-based 3D pipe automation layout with enhanced ant colony optimization algorithm. *Automation in Construction*, **167**, 105689. https://doi.org/10.1016/j.autcon.2024.105689.

Luo, T., Lu, J., Qin, Q., & Liu, Y. (2022). Improved JPS path optimization for mobile robots based on angel-propagation theta* algorithm. *Algorithms*, **15**, 198. https://doi.org/10.3390/a15060198.

Min, J. G., Ruy, W. S., & Park, C. S. (2020). Faster pipe auto-routing using improved jump point search. *International Journal of Naval Architecture and Ocean Engineering*, **12**, 596–604. https://doi.org/10.1016/j.ijnaoe.2020.07.004.

Nandari, M. H., Hejazi, S. R., & Palhang, M. (2015). MCPN, octree neighbor finding during tree model construction using parental neighboring rule. *3D Research*, **6**, 1–15. https://doi.org/10.1007/s13319-015-0060-9.

Nobes, T. K., Harabor, D., Wybrow, M., & Walsh, S. D. C. (2022). The jps pathfinding system in 3d. *Proceedings of the International Symposium on Combinatorial Search* (pp. 145–152). AAAI Press.

Noreen, I. (2020). Collision free smooth path for mobile robots in cluttered environment using an economical clamped cubic B-Spline. *Symmetry*, **12**, 1567. https://doi.org/10.3390/sym12091567.

Park, J. H., & Storch, R. L. (2002). Pipe-routing algorithm development: Case study of a ship engine room design. *Expert Systems with Applications*, **23**, 299–309. https://doi.org/10.1016/S0957-4174(02)00049-0.

Rodenberg, O. B. P. M., Verbree, E., & Zlatanova, S. (2016). Indoor A* pathfinding through an octree representation of a point cloud. *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences* (**vol. 4**, pp. 249–255). Copernicus GmbH.

Samet, H. (1989). Neighbor finding in images represented by octrees. *Computer Vision, Graphics, and Image Processing*, **46**, 367–386. https://doi.org/10.1016/0734-189X(89)90038-8.

Socha, K., & Dorigo, M. (2008). Ant colony optimization for continuous domains. *European Journal of Operation Research*, **185**, 1155–1173. https://doi.org/10.1016/j.ejor.2006.06.046.

Zhou, X., Gui, W., Heidari, A. A., Cai, Z., Liang, G., & Chen, H. (2023). Random following ant colony optimization: Continuous and binary variants for global optimization and feature selection. *Applied Soft Computing*, **144**, 110513. https://doi.org/10.1016/j.asoc.2023.110513.

Zhu, Z., La, R. G., & Van, T. M. J. L. (2017). A methodology to enable automatic 3D routing of aircraft electrical wiring Interconnection System. *CEAS Aeronautical Journal*, **8**, 287–302. https://doi.org/10.1007/s13272-017-0238-3.

Received: May 30, 2024. **Revised:** September 26, 2024. **Accepted:** September 27, 2024