

## RESEARCH ARTICLE

# Automatic Dual Crane Cooperative Path Planning Based on Multiple RRT Algorithm for Narrow Path Finding Scenario

JI-CHUL KIM<sup>1</sup>, HANMIN LEE<sup>1</sup>, YEONGJAE KIM<sup>1</sup>, AND DONGWOOK LEE<sup>2,3</sup> <sup>1</sup>Department of Smart Industrial Machine Technologies, Korea Institute of Machinery and Materials, Daejeon 34103, Republic of Korea<sup>2</sup>Department of Mechanical and Automotive Engineering, Kongju National University, Cheonan 31080, Republic of Korea<sup>3</sup>Global Institute of Manufacturing Technology (GITECH), Kongju National University, Cheonan 31080, Republic of Korea

Corresponding author: Dongwook Lee (dwlee@kongju.ac.kr)


This work was supported in part by the National Research and Development Project for Smart Construction Technology funded by Korea Agency for Infrastructure Technology Advancement under the Ministry of Land, Infrastructure and Transport, and managed by Korea Expressway Corporation under Grant RS-2020-KA157074; and in part by the “Regional Innovation Strategy (RIS)” through the National Research Foundation of Korea (NRF) funded by the Ministry of Education (MOE) under Grant 2021RIS-004.

**ABSTRACT** Dual crane lifting, wherein two cranes collaborate to lift a single workpiece, serves as an essential solution in scenarios in which employing a single, sufficiently large crane is impractical due to cost constraints, ground conditions, and spatial limitations. Due to the complexity of double crane lifting operations, the implementation of automated path generation minimizes the risk of human error and removes the potential for accidents by simulating and validating the generated crane path. We propose a novel multiple rapidly-exploring random trees (RRT) based algorithm designed specifically for dual crane systems to produce lifting paths, particularly in challenging ‘narrow path finding’ scenarios. The multiple RRT method is an efficient way to find paths in environments with high complexity and low connectivity through a strategy that allows new trees to be generated and grown whenever a newly generated node that cannot be connected to an existing tree occurs. The proposed path planning algorithm not only adapts the multiple RRT method to the dual crane systems but also incorporates ideas to enhance the optimality of generated paths while reducing computational time. The effectiveness of this algorithm has been validated through a case studies covering various scenario.

**INDEX TERMS** Dual crane cooperative system, path planning, multiple rapid-exploring random tree.

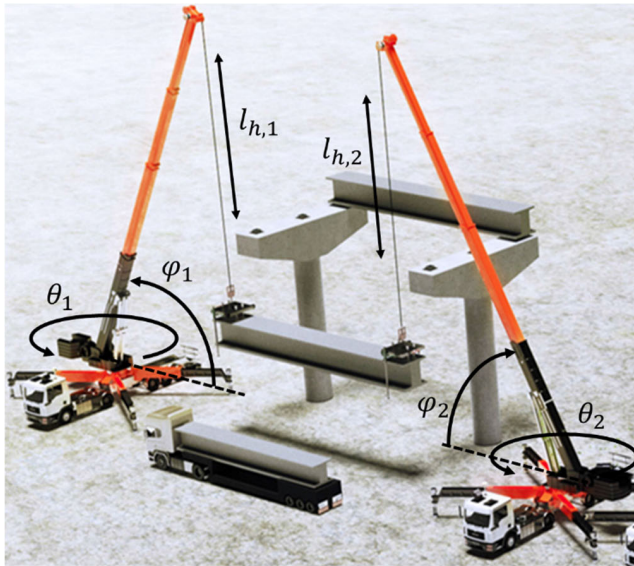
## I. INTRODUCTION

Mobile cranes play an essential role in the construction and civil engineering sectors, serving as indispensable machines for lifting and transporting heavy loads. In particular, dual crane lifting, where two cranes work together to lift and place a single workpiece as shown in Fig. 1, plays an important role in situations where single massive crane cannot be utilized for a variety of reasons including cost considerations, ground conditions insufficient to support the weight of a single large crane, and limited space issues that make access or maneuvering difficult [1], [2], [6], [11], [12]. The dual crane system

The associate editor coordinating the review of this manuscript and approving it for publication was Hao Wang .

also has the added benefit of reducing the yaw swing motion of the workpiece, since there are two anchor points at each end to hold the workpiece.

However, in the case of a double crane lifting operation, the difficulty level for the operator rises dramatically due to the increased complexity of the system [3]. Since two cranes are moving a single object, two operators, one on each crane, need to be precisely synchronized with each other. Throughout the lifting process, the two cranes must move in harmony to ensure that no collisions occur and that the dual crane system remains stable without turnover. In general, the person working inside the crane has limited visual information about the work environment, so collaboration with at least three people including an observer outside the crane is



**FIGURE 1.** Operational environment and kinematics of dual crane cooperative system.

needed. Achieving such complex and synchronized operation manually requires highly skilled workers and well-developed path planning.

Due to the complexity of double crane lifting operations, manual planning of these transfers is often insufficient to ensure safety, precision, and efficiency, making automated path planning an essential requirement. Automated path generation minimizes the risk of human error, which can lead to collision accidents and equipment damage. By simulating and validating generated crane movements, the likelihood of accidents during dual crane lifting operations can be significantly reduced. In addition, generated crane operation path allows even less experienced crane operators to perform complex dual crane lifting tasks safely and efficiently by providing motion guidance on monitor. Finally, as more industrial and construction sites attempt to automate a variety of tasks to reduce labor costs and improve safety and productivity, automated path planning is a fundamental technology that should be at the forefront.

Prior research efforts have predominantly focused on automated path planning for single cranes. In these studies, various methodologies, including heuristic search methods [4], bidirectional expanding trees [5], probabilistic road maps (PRM) [6], A\* search algorithms [7], rapidly-exploring random trees (RRT) [8] and genetic algorithm [9] have been explored to address single crane lifting path generation.

In contrast to the extensive research on single cranes, studies addressing automated path planning for dual crane cooperative lift operations have been relatively limited. In fact, generating paths for dual crane scenarios presents a significantly more complex and challenging problem compared to single crane scenarios [3].

First, dual crane cooperative lifting systems introduce a twofold increase in the degree of freedom (DOF) when

compared to single crane systems. This expanded DOF significantly broadens the solution space that algorithms must explore. (In general, this solution space is referred to as the configuration space (C-space) because the path of the crane is represented by the successive configurations of the crane.) As a result, even when applying similar algorithms, the computational time and efficiency tend to rapidly increase in dual crane scenarios.

Second, collision avoidance is another major challenge in dual crane lifting operations. Given the limited space on construction sites, there is a much higher chance of crane-to-crane, crane-to-workpiece, crane-to-obstacle, and workpiece-to-obstacle collisions. Dual cranes are often used to lift very long or exceptionally heavy objects through limited space, which inherently increases the risk of accidents. Moreover, the dual crane system lifts a single load leading to the formation of a closed chain between the two cranes. This chain is determined by the kinematic equilibrium of the cables and the load, causing extra consideration for modeling for collision check.

Furthermore, the additional horizontal load due to the tilt of the hoist cable on this chain can potentially affect the stability of the crane, causing a turnover in the worst case. Therefore, additional constraints on the tilt of the hoist cable and the lifting object must be considered to exclude this possibility.

Early work on finding paths for dual cranes suggested applying hill climbing and A\* algorithms [10]. The hill-climbing approach could cause the cranes to become stuck in local minima, preventing suitable paths from being obtained. Moreover, the computational time of the A\* algorithm dramatically increases as the environmental complexity increase. To overcome these limitations, genetic algorithms have been used to propose methods to find and optimize candidate paths in the configuration space (C-space) [11]. This approach represented a significant improvement over the previous method. However, it still requires substantial computational resources, resulting in long processing times and inefficiency. Later research proposed a two-step algorithm for both single and double cranes that finds the crane boom's posture path based on probabilistic road maps (PRM) and then find the hoist cable length trajectory [6]. In the above three studies, the kinematics of the crane is simplified by fixing the tilting angles of the hoisting cable and lifting objects to 0 degrees. This oversimplification of the modeling is not representative of real-world scenarios where tilting inevitably occurs during many lifting operations. These excessive constraints on the cable and object angles reduce the overall system's DOF and finding paths in this reduced space is one of the reasons why lower computational load or time have been able to achieve in previous research. In their most recent study, parallel genetic algorithm-based path generation considering the hoist tilting angle and kinematic chain was proposed [12]. In this study, parallel computing using GPUs was utilized to handle the large amount of computation required for path generation. Moreover, relatively simple scenarios were considered to verify this algorithm.

The reason why finding a dual crane lifting path requires a large computational load is not only because of the high DOF of C-space, but also collision avoidance between dual crane system and the construction site, and the tilt constraints of the hoist cable and the lifting object, which significantly reduce the effective area of the solution space and make it difficult to find a feasible path. It becomes more challenging if the construction site is complex with many obstacles, or if the cranes are restricted in their placement due to space constraints.

One of the typical difficult lifting problems in real-world construction site is when the length of the lifting object is similar to or greater than the distance between two cranes, and the object must be transported by passing between them. In this case, it is only possible to pass between the two cranes through a specific twisting maneuver. From the perspective of automatic path generation, it is very difficult to find a path in C-space because the space belonging to the initial configuration and the space belonging to the final configuration are only connected by a very narrow space (corresponding to a specific maneuver). To define lifting cases that find paths through these kinds of narrow spaces, this paper will use the term ‘narrow path finding’ scenario.

In this paper, we propose an algorithm that can generate appropriate dual crane cooperative lifting paths in various scenarios, including the challenging narrow path finding case, which has not been considered in the previous research. The proposed algorithm utilizes the Multiple RRT method considering the characteristics of dual crane system to generate lifting paths with optimality and computational efficiency in high dimensional and low connectivity C-space.

In section II, a base problem formulation is provided to describe the problem definition, dual crane modeling, and the feasibility check for a given configuration of a dual crane. In section III, we describe the proposed multiple RRT-based path planning algorithm in detail. In section IV, the performance of the suggested methodology is validated through case studies involving various scenarios. Finally, in section V, the summary of this paper is provided, and directions for future research are outlined.

## II. PROBLEM FORMULATION

In this section, the modeling of the dual crane system which serves as the foundation for automated path generation and collision detection is describe. Afterward, we will explain how to solve the hoist cables-object suspension problem that arises during the modeling process. Furthermore, it also explains the feasibility check process to verify whether the candidate configuration of the dual crane is a valid or not.

### A. DUAL CRANE SYSTEM MODELING

In this study, it is assumed that the scenario of lifting girders on a pier or abutment for the construction of a bridge is the operational environment of a dual crane system as shown in Fig. 1.

Each crane has 3 degrees of freedom, as illustrated in Fig. 1: the swing angle ( $\theta_i$ ), luffing angle ( $\varphi_i$ ), and hoist length ( $l_{h,i}$ ).

Basically, the length of the boom,  $l_{b,i}$  is adjustable. However, once the workpiece is connected and the load is applied, the length of the boom remains fixed throughout the operation.

The basic information given for automatic path planning is the initial configuration where the girder is connected to the crane, the final configuration at the girder’s destination, the boom length, the operation range for each crane, and the location and size of obstacles within the worksite in the form of boxes.  $i$ -th obstacles are defined using the coordinate of the two opposite endpoints of the box as follows

$$Obstacle_i = (x_1, y_1, z_1, x_2, y_2, z_2)_i \quad (1)$$

There are various shapes and sizes of obstacles in construction sites, but even with complex shapes, it is necessary to maintain a sufficient distance from the obstacle during transportation to ensure collision avoidance. Therefore, it can be modeled as a box with a sufficient margin from the actual obstacle for path generation.

Additionally, the construction site is assumed to be static, meaning there is no change in the obstacles while the dual cranes are working.

As mentioned above, because the length of the boom is fixed during the work process, so each crane maintains a 3-degree-of-freedom system during operation. Therefore, the generated paths will occur in a 6 DOFs configuration space (C-space). The dual crane’s path, denoted as  $Path_{dual}$ , can be represented as a set of 6-DOF configurations  $C_i$  as follows.

$$C_i = (\theta_1, \varphi_1, l_{h,1}, \theta_2, \varphi_2, l_{h,2})_i \in Path_{dual}, i = 1, 2, \dots, n \quad (2)$$

Here,  $C_1$ , and  $C_n$  represent the initial and final configurations, and  $C_{i+1}$  is the next configuration after  $C_i$ .

### B. SOLVE SUSPENSION PROBLEM

To simulate crane operations, and verify collision occurrences, it is necessary to solve the suspension system problem created by closed chain of two hoist cables and the lifting target. Due to this suspension system, even though the crane’s 6-DOFs configuration is determined, an additional 2- DOFs that represent the inclinations of each hoist cable need to be considered in the dual crane model. Furthermore, the lifting object also has a 3-DOFs in position and 2-DOFs in inclination as it is suspended in the air. This suspension system is unique to dual crane systems and does not occur with single crane that does not form a closed chain.

These additional degrees of freedom introduced by the hoist cable and lifting object inclination are uniquely determined by kinematic constraints and equilibrium conditions for given a crane configuration. To address the suspension subsystem, we utilize some physical insights.

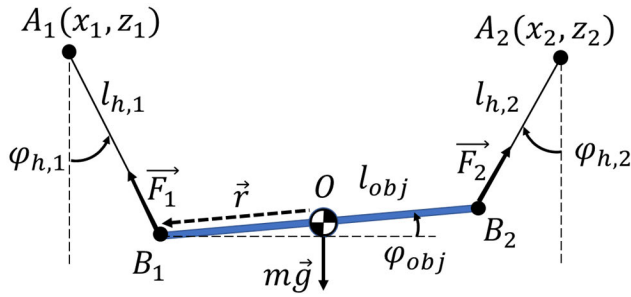


FIGURE 2. Suspension subsystem in dual crane system.

The first insight is that once the configuration of the dual crane is determined and the suspension system reaches an equilibrium state, the hoist cables and the center of gravity of the lifting object will lie in a vertical plane encompassing the boom ends of both cranes. Otherwise, the system will oscillate because of gravity. Therefore, the equilibrium position of the suspension system can be analyzed in a 2D plane as shown in Fig. 2.

In Fig. 2,  $A_1$  and  $A_2$  represent the boom ends,  $B_1$  and  $B_2$  represent the attach anchor between the cable and the object, and the slope of the cable and the object on the plane is defined as  $\varphi_{h,i}$  ( $i = 1, 2$ , tilt angles between the hoist cable and the z-axis across  $A_i$ ) and  $\varphi_{obj}$  (slope angle between the line across the attach anchors and x-axis across  $B_1$ ), respectively. The length of the hoist cable and the distance between the attach anchor are denoted by  $l_{h,1}$ ,  $l_{h,2}$ , and  $l_{obj}$ , respectively.

The lifting object is assumed to be a long and constant material object such as a girder used for bridge construction, and the location of the attach anchor is also assumed to be installed at both ends of the girder. Additionally, since the material of the object is assumed to be uniform, the center of mass  $O$  is located in the middle of the girder, as shown in the Fig. 2. Finally, the tension on each cable is represented by the  $\vec{F}_1$  and  $\vec{F}_2$  vectors, respectively.

The three factors that determine the final state of a closed chain structure are 1) the geometric closed chain, 2) the equilibrium of forces, and 3) the equilibrium of torques. These are represented by the following equations.

$$x_1 + l_{h,1} \sin(\varphi_{h,1}) + l_{obj} \cos(\varphi_{obj}) + l_{h2} \sin(\varphi_{h,2}) = x_2 \quad (3)$$

$$z_1 - l_{h,1} \cos(\varphi_{h,1}) + l_{obj} \sin(\varphi_{obj}) + l_{h,2} \cos(\varphi_{h,2}) = z_2 \quad (4)$$

$$|\vec{F}_1| \cos(\varphi_{h,1}) + |\vec{F}_2| \cos(\varphi_{h,2}) = m\vec{g} \quad (5)$$

$$|\vec{F}_1| \sin(\varphi_{h,1}) = |\vec{F}_2| \sin(\varphi_{h,2}) \quad (6)$$

$$\vec{r} \times \vec{F}_1 - \vec{r} \times \vec{F}_2 = 0 \quad (7)$$

In the equation above,  $m$  is the weight of the girder,  $\vec{g}$  is the gravity vector, and  $\vec{r}$  is the position vectors of  $B_1$  relative to the  $O$ .

- 1:  $\varphi_{h,1} = -\varphi_{h,max} : \varphi_{h,int} : \varphi_{h,max}$
- 2: **for all**  $\varphi_{h,1}$  **do**
- 3:   calculate  $\varphi_{h,2}(i)$  from  $\varphi_{h,1}(i)$
- 4:   calculate position of  $O(i)$  from  $\varphi_{h,2}(i)$ ,  $\varphi_{h,1}(i)$
- 5: **end for**
- 6: **if**  $O(i)$  is local minimum
- 7:   **return**  $\varphi_{h,1}(i)$ ,  $\varphi_{h,2}(i)$ ,  $\varphi_{obj}(i)$
- 8: **elseif**  $O$  has no local minimum
- 9:   **return** no feasible config.
- 10: **end if**

FIGURE 3. Pseudo-code for finding equilibrium state of suspension system.

Since the number of unknown variables in (3)-(7) ( $\varphi_{h,1}$ ,  $\varphi_{h,2}$ ,  $\varphi_{obj}$ ,  $|\vec{F}_1|$ ,  $|\vec{F}_2|$ ) is the same as the number of equations, we can determine the value of each variable by solving this system of equations. However, due to the highly nonlinear characteristics of these equations, it is difficult to obtain an analytic solution. While numerical methods such as using the GNU Scientific Library [26] can be used to find solutions, a simpler approach is used in this study by taking advantage of the limited tilting range of the hoist cable and the following physical insight: among the many possible solutions of the suspension system, the equilibrium state occurs when the center of gravity of the girder is at its lowest point.

This method involves dividing  $\varphi_{h,1}$  into sufficiently small resolution intervals within the maximum allowable range  $[-\varphi_{h,max}, \varphi_{h,max}]$ , for each case, determining  $\varphi_{h,2}$  and the position of the center of gravity  $O$  using (3) and (4). Then, it checks whether the position of  $O$  has a local minimum within this range. If there is a local minimum, then  $(\varphi_{h,1}, \varphi_{h,2})$  in that case is the equilibrium point. If there is no local minimum, the given configuration is invalid because there are no equilibrium points within the allowable range for  $\varphi_{h,1}$ . The pseudo-code to find the equilibrium state of the suspension system and the formula to determine  $\varphi_{h,2}$  when  $\varphi_{h,1}$  is given are presented in Fig. 3 and Appendix, respectively.

### C. FEASIBILITY CHECK OF THE DUAL CRANE CONFIGURATION

Once the dual crane system, including the suspension subsystem, is determined from the given crane configuration, it becomes possible to determine whether this configuration is feasible to satisfy the given constraints. For a candidate configuration  $C$  to be feasible, it must satisfy the following three conditions:

- 1) Is each argument of  $C$  within the lower and upper bounds for each crane?
- 2) Is the distance between the boom ends not too far to realize a suspension subsystem? Additionally, is the hoist cable tilt angle of each crane within a range that does not compromise the stability of the dual crane system?

```

1: Solve suspension problem for C
2: if C is not within upper & lower bound
3:   C is not feasible
4: elseif (distance between  $A_1$  and  $A_2$ )
      > ( $l_{h,1} + l_{h,2} + l_{obj}$ )
5:   C is not feasible
6: elseif ( $\varphi_{h,1} > \varphi_{h,max}$ ) or ( $\varphi_{h,2} > \varphi_{h,max}$ )
      or ( $\varphi_{obj} > \varphi_{obj,max}$ )
7:   C is not feasible
8: elseif any collisions occur
9:   C is not feasible
10: else
11:   C is feasible
12: endif

```

FIGURE 4. Pseudo-code for feasibility check for configuration C.

3) Does any element of the dual crane system collide with each other or with obstacles?

For a given candidate configuration, it is verified in the order from 1) to 3), and if any of the conditions are not satisfied in the middle, the verification process is stopped and the configuration is determined to be infeasible. Fig. 4 shows a pseudo code of how feasibility check works.

In the first constraint, the lower bound  $C_{min}$  and upper bound  $C_{max}$  are determined by the specification of the crane and may have different limits in the case that the types of the two cranes are different. The range of the luff angle depends on the load weight on the hoist cable and boom length.

In the second step, two verifications are performed. First, the boom ends are checked to determine whether distance between them are too far apart such that it is greater than the combined length of the hoist cables and the lifting object. If this condition is met, the configuration is infeasible. Next, the calculated tilting angle of the hoist cables ( $\varphi_{h,1}$ ,  $\varphi_{h,2}$ ) and lifting object ( $\varphi_{obj}$ ) at the equilibrium condition are verified to determine whether they are less than the specified thresholds,  $\varphi_{h,max}$  and  $\varphi_{obj,max}$ . Setting appropriate thresholds,  $\varphi_{h,max}$ , and  $\varphi_{obj,max}$ , is critical for the efficiency and stability of path generation. Insufficient or overly loose constraints may result in an excessive lateral force and moment loads on the dual crane system, potentially leading to turnover accidents. On the other hand, overly conservative constraints also make it difficult to find a solution. Additionally, since swing motion can occur while implementing the path generated for a dual crane system, it is advisable to set a slightly conservative threshold, especially for  $\varphi_{h,max}$ . Since the range of  $\varphi_{h,1}$  is already verified during the process of solving the suspension subsystem, we only need to verify the calculated  $\varphi_{h,2}$  and  $\varphi_{obj}$ .

In the final step, the collision check is performed for all combinations of each crane's boom, lifting object, hoist cable, and obstacles to ensure that no collisions occur. Robotics system toolbox from the MATLAB was utilized to implement the dual crane system and perform collision verification.

```

1: do while PATH not found
2:   NEW_NODE generate
3:   find NEAREST_NODE
4:   collision check
4:   if no collision
5:     connect to TREE
6:   end if
7: end do

```

FIGURE 5. Pseudo-code of RRT algorithm.

The task of dual crane path planning is to find the optimal path  $Path_{dual}^*$  connecting the initial and final configurations without collision based on the crane model and problem foundation given above.

### III. PROPOSED METHOD

#### A. MULTIPLE RRT METHOD

RRT and PRM are two representative types of sampling-based path planning algorithms commonly used in robotics for motion and path planning when obstacles are present.

Regarding PRM, a roadmap is constructed by randomly sampling the nodes across the entire C-space and connecting all feasible nodes and feasible edges between them. Then, this roadmap is utilized to explore all possible paths from the starting point to the goal, and finally, the path with the lowest cost is selected [13], [14].

The preprocess of sampling all C-space in the PRM is not well suited to the dual-crane lifting path problem because in the high-dimensional dual-crane C-space of 6DOF, the number of samples that need to be generated and validated becomes too large. In addition, in a narrow path finding scenario, an exhaustive search method that generates a roadmap for this entire space is inefficient.

On the other hand, the RRT algorithm is suitable for finding paths in higher-dimensional and more complex environments. Fig. 5 shows a pseudo code of how basic RRT works [19]. RRT operates by randomly sampling new nodes within the C-space and checking the connectivity of this newly generated node with the existing tree. This process is repeated to expand the tree until it reaches the end point from the start point.

The main advantages of RRT over PRM for finding dual crane cooperative path planning in high DOF C-space is that it does not require preprocessing of the environment [15], [16], [17], [18]. Another advantage is that there is fewer parameters to be tuned for achieving desired performance, in contrast to PRM, where tuning the parameter that indicates how much to sample in C-space has a significant impact on the efficiency and performance of the algorithm.

However, the basic RRT algorithm still exhibits limited efficiency in finding paths quickly in complex environments [19]. In particular, for the 'narrow path finding' case,

```

1: load Scenario_info
2: do while PATH not found
3:   do while NEW_NODE is feasible
4:     NEW_NODE generate
5:   end do
6:   for each TREE do
7:     find NEAREST_NODE
8:     find NEAREST_EDGE
9:     if (distance to NEAREST_NODE
          > distance to NEAREST_EDGE)
10:      try to connect to NEAREST_NODE
11:    else
12:      try to connect to NEAREST_EDGE
13:    end if
14:  end for
15:  if connect to more than 2 TREES
16:    merge all TREES connected to NEW_NODE
17:  elseif connect to no TREE
18:    create NEW_TREE from NEW_NODE
19:  end if
20: end do
21: for  $i=1 : N_{Smooth}$  do
22:   smooth the PATH
23: end for

```

FIGURE 6. Pseudo-code for Main Function of multiple RRT based dual crane path planning algorithm.

when extending a single tree from the starting node, nodes generated in spaces separated from the starting node and only connected by narrow doors are often discarded because they have a low probability of being connected to the tree. This results in a very slow expansion rate.

The multiple RRT method overcomes these limitations of basic RRT and provides an efficient way to find paths in environments with high complexity and DOF [19]. Multiple RRT is implemented through a simple but effective strategy that allows new trees to be created and grown whenever newly generated node that cannot be connected to an existing tree occurs. Multiple RRT allows multiple trees to grow in separate spaces. As each tree grows in its respective space and eventually generates connection points, it efficiently creates a path from the starting node to the final node. Therefore, these characteristics of multiple RRT make it an efficient method for dual crane path planning.

### B. MULTIPLE RRT-BASED DUAL CRANE PATH PLANNING

The pseudo-code of the proposed multiple RRT-based dual crane cooperative lifting path planning algorithm is shown in Fig. 6.

The proposed path planning algorithm, summarized by the pseudocode above, is not only an adaptation of the multiple RRT method proposed in previous research to the dual crane system but also integrates two ideas to improve performance.

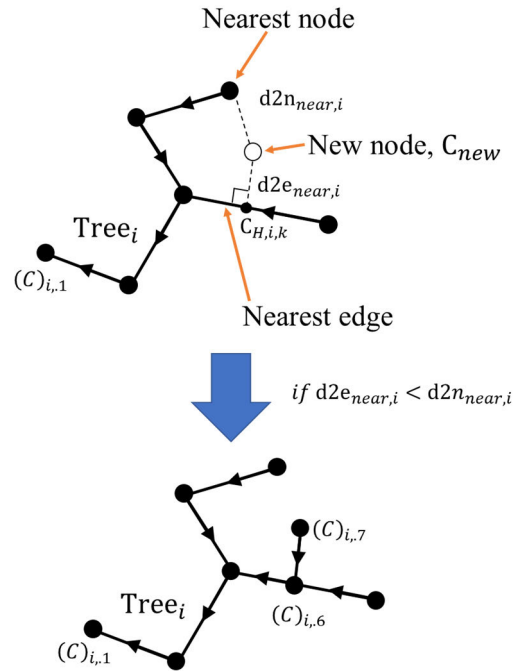


FIGURE 7. Schematic of new node connection algorithm.

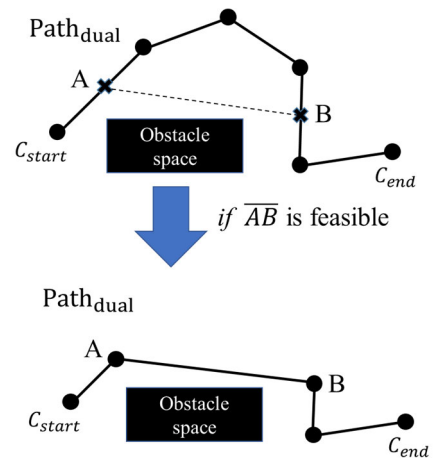


FIGURE 8. Working principle of the smoothing algorithm.

The first idea is that when newly generated nodes perform a connectivity check on each tree, they not only try to connect to the nearest node but also check the possibility of connecting to the nearest edge. It connects to the closest node or the closest edge, whichever is closer. This approach not only increases the likelihood that the connection will succeed but also allows the tree to expand more optimally in C-space. Here, a more optimal tree means that the path created between two nodes is shorter in C-space. This concept is illustrated schematically in Fig. 7 for a 2-DOF space.

The second idea is to improve the optimality of paths generated by multiple-RRT. Typically, paths generated by RRT may not take the shortest route in the C-space, resulting in generating suboptimal path. This characteristic of RRT led

to the development of an improved algorithm called RRT\* which aims to make the tree more optimal by modifying tree at each iteration by minimizing the lengths from the start node to each endpoint of the tree [20], [21]. However, modifying the tree for every iteration is inefficient because it constantly consumes computation [22]. In addition, in the case of multiple RRT, this approach of RRT\* cannot be applied because some trees do not contain starting and final nodes. Therefore, to ensure the optimality of the generated path, a smoothing algorithm is added after the path is once generated from multiple RRT method. The smoothing algorithm works by selecting two edges from the generated path, randomly selecting two nodes in the middle of each edge, and replacing the original suboptimal path between the two nodes with a direct path in C-space when the configurations between the two nodes becomes feasible. By repeating the same operation as many times as necessary, it is possible to optimally change the generated dual crane path. The schematic principle of the smoothing algorithm can be seen in Fig. 8.

In the remainder of this section, we describe in detail how multiple RRTs and the ideas described above are applied to the dual crane lifting path planning algorithm.

**The initialization (Line 1 in Fig. 6)** of the Multiple RRT based path generation algorithm starts with loading information about the operation scenario. The scenario information includes the location of each crane base and boom length, initial configuration  $C_{start}$ , final configuration  $C_{end}$ , length of boom length  $l_{b,i}$ , length of lifting object  $l_{obj}$ , lower and upper range of the operation  $C_{min}$ ,  $C_{max}$ , and location and size of the obstacle  $Obstacle_i$ .

Once basic information regarding the dual crane operation is obtained, two trees, each containing a start node and an end node each, are generated.

Tree<sub>*i*</sub> is defined as a set of edges  $E_{k,i}$ , as shown in the following Equation.

$$E_{k,j} \in Tree_i (i = 1, 2, \dots, n_{tree}, k = 1, 2, \dots, n_{edge,j}) \quad (8)$$

*i* is an index number to distinguish each tree and is given according to the order in which the trees were created. Similarly, each  $E_{k,i}$  belonging to Tree<sub>*i*</sub> has an index number *k* and is given according to the order in which the edges were connected. The total number of trees and the total number of edges in each tree are defined by  $n_{tree}$  and  $n_{edge,i}$ .

$E_{k,i}$  consists of a pair of nodes  $C$  and the number  $n_{parent}$ , which is index of edge containing the parent node of  $C$ , as shown below.

$$E_{k,i} = (C, n_{parent})_{k,i} \quad (9)$$

The parent node specifies the node to which the new node is attached when it is added to the tree. If the 7th edge added to Tree<sub>1</sub> is given as  $E_{1,7} = (C, 3)_{1,7}$ , it means that the node  $(C)_{1,7}$  belonging to  $E_{1,7}$  is connected to  $(C)_{1,3}$  belonging to  $E_{1,3}$ . The first node that forms each tree has no parent node. Thus,  $n_{parent}$  is set to 0. The relationship between parent and son nodes is also shown by the arrows in Fig. 7.

Tree<sub>1</sub> and Tree<sub>2</sub> can be created with the start node and final node as follows.

$$Tree_1 = \{(C_{start}, 0)_{1,1}\} \quad (10)$$

$$Tree_2 = \{(C_{end}, 0)_{1,2}\} \quad (11)$$

After initialization process, including loading scenario information and generating initial trees, are completed, iteration for path generation begins. Each iteration consists of 1) New node generation, 2) Finding Nearest node & Edge of each Tree, 3) Connect to Trees, 4) Combine Trees, and 5) Check for complete path.

**The first step (Line 3-5 in Fig. 6)** of the iteration is the generation of a new node  $C_{new}$ . The swing angle and boom angle of the new node  $C_{new}$  are randomly generated between  $C_{min}$  and  $C_{max}$ . The hoist length is randomly generated between  $C_{min}$  and the height of the boom end which is determined by the boom length and boom angle for each crane as shown below.

$$C_{new} = (\theta_1, \varphi_1, l_{h,1}, \theta_2, \varphi_2, l_{h,2})_{new} \quad (12)$$

$$(\theta_i)_{new} = \theta_{i,min} + (\theta_{i,max} - \theta_{i,min}) * rand(0, 1) \quad (13)$$

$$(\varphi_i)_{new} = \varphi_{i,min} + (\varphi_{i,max} - \varphi_{i,min}) * rand(0, 1) \quad (14)$$

$$l_{h,i,max}^* = \min(l_{h,i,max}, l_{boom,i} * \cos((\varphi_i)_{new})) \quad (15)$$

$$l_{h,i} = l_{h,i,min} + (l_{h,i,max}^* - l_{h,i,min}) * rand(0, 1) \quad (16)$$

where,  $rand(0, 1)$  is a uniformly distributed random number generated between 0 and 1.

The generated  $C_{new}$  is verified by the feasibility check algorithm presented in III-C to ensure that it is a valid configuration. If it passes the feasibility check, the algorithm goes to the next step with  $C_{new}$ ; if it fails, a new  $C_{new}$  is randomly generated and repeated until a valid configuration is obtained.

**The second step (Line 6-8 in Fig. 6)** is to find the closest node and edge to all the trees that have been created thus far. For each Tree<sub>*i*</sub>, the distance in C-space between each node  $(C)_{i,k}$  of Tree<sub>*i*</sub> and  $C_{new}$  is computed in order and find the node with the shortest distance. the order index *k* and the distance from  $C_{new}$  for the nearest node in the Tree<sub>*i*</sub> are stored in  $N_{near,i}$  and  $d2n_{near,i}$ , respectively. (Refer to Fig. 7).

Additionally, we need to find the nearest edge from Tree<sub>*i*</sub> using the distance from  $C_{new}$  to  $E_{i,k}$ , which can be calculated by the distance from  $C_{new}$  to the perpendicular foot  $C_{H,i,k}$  on Edge  $E_{i,k}$ . The relationship between  $C_{new}$ ,  $C_{H,i,k}$ ,  $E_{i,k}$  and distance  $d2e_{i,k}$  can be seen in Fig. 7. After distance from  $C_{new}$  to each edge is calculated, the order index *k* of the nearest edge in the Tree<sub>*i*</sub> is stored in  $E_{near,i}$ , and the distance is stored in  $d2e_{near,i}$ . If the perpendicular foot  $C_{H,i,k}$  is out of range of the edge  $E_{i,k}$  and is on the extension line of edge, it is an invalid case. In this case, infinity value is assigned to  $d2e_{near,i}$  for the edge not to be considered as an option when  $C_{new}$  attempts to connect to Tree<sub>*i*</sub> in next step. The formula to find the distance  $d2e_{i,k}$  between  $C_{new}$  and the perpendicular foot  $C_{H,i,k}$  on Edge  $E_{i,k}$  is as follows.

$$C_{H,i,k} = (C)_{i,k} + \vec{E}_{i,k} \left( \vec{E}_{i,k} \cdot \overrightarrow{(C)_{i,k} C_{new}} \right) / \left| \vec{E}_{i,k} \right| \quad (17)$$

$$\vec{E}_{i,k} = (C)_{((n_{parent})_{i,k,j})} - (C)_{i,k} \quad (18)$$

$$\overrightarrow{(C)_{i,k} C_{new}} = C_{new} - (C)_{i,k} \quad (19)$$

$$d2e_{i,k} = |C_{new} - C_{H,i,k}| \quad (20)$$

where,  $\vec{E}_{i,k}$  is vector notation of  $E_{i,k}$  in C-space.

As a third step (Line 9-13 in Fig. 6), for each tree,  $C_{new}$  attempts to connect to the nearest node or the nearest edge. If the distance to the nearest node is shorter ( $d2n_{near,i} < d2e_{near,i}$ ), it must be verified that  $C_{new}$ , and nearest node  $(C)_{i,N_{near,i}}$  can be connected as a new edge. The feasibility of connecting an edge is determined by performing a feasibility check on each  $C_{interp,m}$  generated by interpolating  $C_{new}$  and  $(C)_{i,N_{near,i}}$  with the desired number of steps.

$$C_{interp,m} = C_{new} + \left( (C)_{N_{near,j,j}} - C_{new} \right) * m/n_{interp}, \quad m = 0, 1, 2, \dots, n_{interp} \quad (21)$$

When an interpolated set of nodes  $C_{interp,m}$  is generated to verify the availability of edge between two nodes, number of interpolation steps  $n_{interp}$  is determined by dividing the average travel distance that occurs at each hoist anchor on the girder ( $B_1$  and  $B_2$  in Fig. 2) by the standard step length  $d_{step}$ . This set of nodes  $C_{interp,m}$  is used to check whether collision occurs during the continuous transport between the configuration nodes.

$d_{step}$  is a parameter that determines how tightly the edge verification is performed in C-space. If  $d_{step}$  is too large, it may not detect collisions that may occur even if all  $C_{interp,m}$  are feasible, and if it is too small, the computation time will increase. The appropriate value for  $d_{step}$  can be determined by simulating with  $C_{interp,m}$  to see if the path between two nodes can be validated with enough density to avoid missing possible collisions.

If any of the  $C_{interp,m}$  do not satisfy feasibility during the validation process, then  $C_{new}$  cannot connect to the nearest node of  $Tree_i$ . If all  $C_{interp,m}$  are valid,  $C_{new}$  is added to  $Tree_i$  with following process:

$$n_{edge,i} \leftarrow n_{edge,i} + 1 \quad (22)$$

$$Tree_i \leftarrow Tree_i \cup \left\{ (C_{new}, N_{near,i})_{n_{edge,i}} \right\} \quad (23)$$

If the distance to the nearest edge is smaller ( $d2n_{near,i} > d2e_{near,i}$ ), the connection to the nearest edge is attempted. To check the feasibility of connecting to the nearest edge,  $C_{interp,m}$  is also generated by interpolating  $C_{new}$  to  $C_{H,i,k}$  evenly using the  $d_{step}$  parameter, and all  $C_{interp,m}$  is passed through the feasibility check process. If any of the  $C_{interp,m}$  is not feasible during the validation process,  $C_{new}$  and the nearest edge cannot be connected. If all  $C_{interp,m}$  are valid, then  $C_{new}$  and  $Tree_i$  are connected as two new edges are added to  $Tree_i$  with following process:

$$n_{edge,i} \leftarrow n_{edge,i} + 1 \quad (24)$$

$$Tree_i \leftarrow Tree_i \cup \left\{ (C_{H,i,k}, (n_{parent})_{E_{near,i,i}})_{n_{edge,i}} \right\} \quad (25)$$

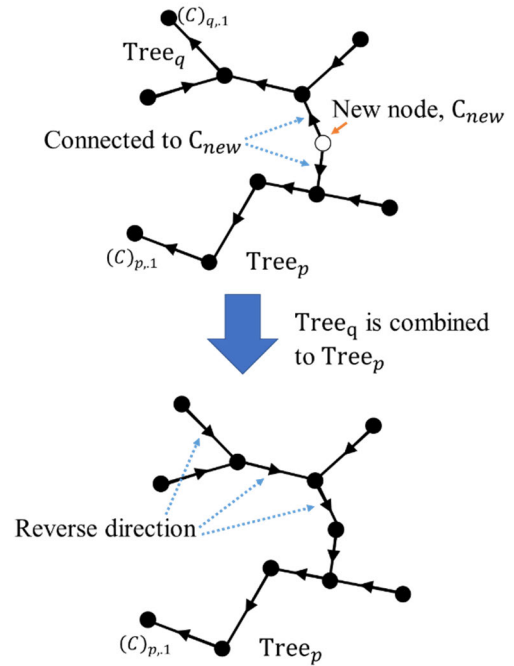


FIGURE 9. Schematic of tree combining algorithm.

$$n_{edge,i} \leftarrow n_{edge,i} + 1 \quad (26)$$

$$Tree_i \leftarrow Tree_i + \left\{ (C_{New}, n_{edge,i} - 1)_{n_{edge,i}} \right\} \quad (27)$$

If  $C_{new}$  fails to connect to both the closest node and the closest edge (Line 17-18 in Fig. 6), it will add  $C_{new}$  as the starting point of a new tree with following process:

$$n_{tree} \leftarrow n_{tree} + 1 \quad (28)$$

$$Tree_{n_{tree}} \leftarrow \left\{ (C_{new}, 0)_{n_{tree,1}} \right\} \quad (29)$$

If  $C_{new}$  is connected to more than one tree, it moves on to the fourth step (Line 15-16 in Fig. 6), in which the trees are combined. In this process, based on the tree with the smallest index value, the remaining trees should be connected via  $C_{new}$  to form a single, larger tree.

If the smallest indexed tree to which  $C_{new}$  is connected is called  $Tree_p$  and the second smallest indexed tree is called  $Tree_q$ , then all the nodes in  $Tree_q$  are added to  $Tree_p$  and the parent node number  $n_{parent}$  is modified for the newly combined nodes.

For the nodes included in the path from the first node  $(C)_{q,1}$  of  $Tree_q$  to  $C_{new}$ , the direction of the parent and son nodes should be reversed in the graph. For the remaining nodes of  $Tree_q$ , the same parent node relationship should be maintained in the graph. Then, the change in the edge index needs to be reflected as it is added to  $Tree_p$ . The schematic of the combining tree sub-algorithm is summarized in Fig. 9.

The final step (Line 2 in Fig. 6) of the iteration is to determine if a tree exists that contains both the start node,  $C_{start}$  and the end node  $C_{end}$ . If such a tree exists, the iteration process terminates, and stores path from  $C_{start}$  to  $C_{end}$  in the  $Path_{dual}$  by repeatedly finding the parent node from the  $C_{end}$ .



The path found in the above iteration process is a path without collision in C-space, but the generated path is likely to be a sub-optimal path with a jig-jagged shape that does not go the shortest distance between two nodes. To improve this feature, we add a postprocessing process called the smoothing algorithm to the generated Path<sub>dual</sub>.

In the smoothing process (Line 21-23 in Fig. 6), first, two edges belonging to the path are randomly selected. Then, for each edge, we randomly select two points A and B on the middle of each edge. Subsequently, a feasibility check is performed on the edge between A and B. If the edge between A and B is feasible, the original path between A and B are discarded and replaced edge between A and B. The schematic for the smoothing process is shown in the Fig. 8. After repeating the smoothing process, a certain number of times, you will finally get the final optimal path, Path<sub>dual</sub><sup>\*</sup>.

IV. CASE STUDY

To verify the performance of the proposed multiple RRT based dual crane path planning algorithm, the algorithm was applied to the 3 scenarios. Each scenario is selected to verify that the proposed algorithm can work well in various cases that may occur at construction sites. The base information of each scenario can be found in Table 1.

The dual-crane system and generated path are simulated using MATLAB’s Robotics system toolbox. Each crane is simplified to a base, boom, and hoist cable, and each element is modelled as a rectangular cylinder. A hoist cable is attached to both ends of the girder, and the closed suspension system of the hoist cable and girder is organized to maintain a kinematic equilibrium condition. Finally, the Obstacle was implemented in the form of a box, colored in orange. The simulation of the initial configuration, final configuration and obstacles for each scenario can be shown in Fig. 10.

In each scenario, the proposed algorithm was applied 30 times to generate 30 paths, and the results show that all attempts generate a dual crane path that can move smoothly from the C<sub>start</sub> to C<sub>end</sub> without collision with obstacles. The Fig. 11 shows a simulation of one of the paths generated in each scenario.

Scenario 1 corresponds to the simplest scenario, where a girder is lifted from the ground to its final position on a bridge pier without obstacles. Scenario 2 adds two obstacles of different heights from Scenario 1 and adjusts to have different y-axis coordinates, with the goal of avoiding the obstacles and lifting to a final position on the other side of the obstacle. The path generation results in the Fig. 11(a) and (b) show that the dual crane cooperative path is successfully generated with a minimum travel distance without collision. Scenarios 1 and 2 are representative of the general case of dual crane lifting on a construction site.

In Scenario 3, the length of the girder and the distance between the two crane bases are set to be the same. Thus, it is difficult for girder to pass between the two cranes and move to the opposite side with normal maneuvers. Such cases can be occurred on construction sites where space is very limited.

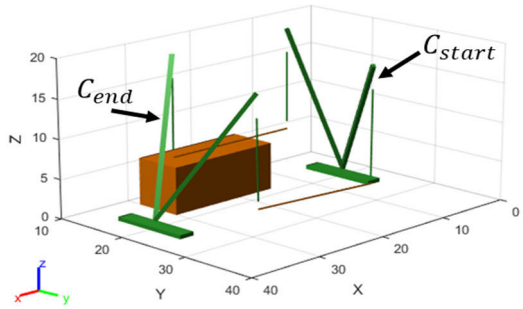
TABLE 1. Base information for each scenario.

Input variables	Scenario 1	Scenario 2	Scenario 3
Base location for crane <i>i</i> , [X, Y, Z] <sub><i>i</i></sub>	[5, 20, 0] <sub>1</sub> [35, 20, 0] <sub>2</sub>	[5, 22, 0] <sub>1</sub> [35, 17, 0] <sub>2</sub>	[0, 20, 0] <sub>1</sub> [35, 20, 0] <sub>2</sub>
Initial config., C <sub>start</sub>	[-30, 30, 55, 55, 15, 15]	[-50, 25, 50, 45, 15, 14]	[0, 0, 60, 60, 23, 23]
Final config., C <sub>end</sub>	[-120, 120, 70 70, 13, 13]	[-115, 90, 55, 75, 11, 13.5]	[-130, 130, 65, 65, 16, 16]
Boom length, <i>l<sub>b,i</sub></i>	<i>l<sub>b,1</sub>, l<sub>b,2</sub></i> = 20	<i>l<sub>b,1</sub>, l<sub>b,2</sub></i> = 20	<i>l<sub>b,1</sub>, l<sub>b,2</sub></i> = 27
Girder length, <i>l<sub>obj</sub></i>	18	15	35
Obstacle, <i>i</i>	[15, 19, 0, 30, 15, 6]	[15, 18, 0, 30, 15, 5] [20, 22, 0, 25, 20, 9] [15, 26, 0, 30, 24, 6]	[9, 2, 0, 10, 22, 8] [47, 2, 0, 48, 22, 8]
C <sub>min</sub> , C <sub>max</sub> φ <sub><i>h,max</i></sub> , φ <sub><i>obj,max</i></sub>	C <sub>min</sub> = [-180, 0, 1], C <sub>max</sub> = [180, 80, <i>l<sub>b</sub></i> ] φ <sub><i>h,max</i></sub> = 10deg, φ <sub><i>obj,max</i></sub> = 10deg		
<i>d<sub>step</sub></i> , <i>n<sub>smooth</sub></i>	<i>d<sub>step</sub></i> = 1, <i>n<sub>smooth</sub></i> = 20		

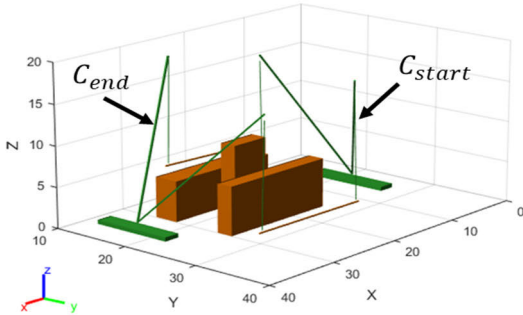
In the Fig. 11(c), the generated path shows that it is only possible to pass between the two cranes by twisting girder with specific maneuver. Since this particular maneuver acts as a very narrow door between the space containing the C<sub>start</sub> and the space containing the C<sub>end</sub> in C-space, scenario 3 can be considered as a “narrow path finding” case. Thus, it is verified that the proposed algorithm performs well in path generation for both general case and the narrow path finding case.

Fig. 12(a) shows the trees and C<sub>new</sub> in C-space at the last iteration when path in Fig. 11(c) for Scenario 3 is generated. This is also the moment just before the tree containing C<sub>start</sub> and the tree containing C<sub>end</sub> are connected and combined through C<sub>new</sub> in C-space. To distinguish between different trees, the tree is represented by different colors. Since the C-space of a dual crane has 6 DOFs, it is impossible to visualize it in three-dimensional space, so in Fig. 12, only the configuration for Crane 1 is represented in three-dimensional space to give an indirect observation of how path is generated in C-space. As shown in Fig. 12(a), it can be seen that each tree has grown without being connected because space containing C<sub>start</sub> and the space containing C<sub>end</sub> are separated from each other and only connected by a narrow door. By repeating the iteration, trees are grown, created, and merged until a C<sub>new</sub> is created that can connect the trees through this narrow door. Such C<sub>new</sub> allows the trees to merge into a single tree that contains both C<sub>start</sub> and C<sub>end</sub>, forming the path shown in the red line in Fig. 12(b).

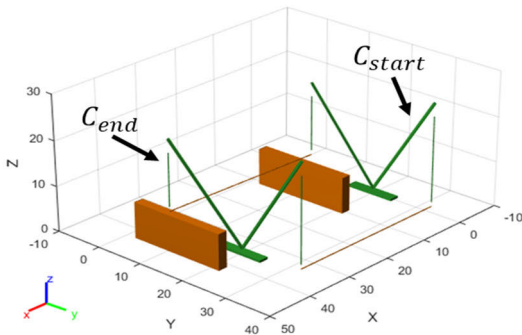
As shown in Fig. 12(b), generated raw path has a zigzag shape in C-space, and when simulating the generated path in real space, it can be seen that the girder is transported with a very inefficient movement. This suboptimal path increases the overall travel distance, which in turn increases the time required for the operation. Therefore, it is necessary to apply a smoothing process so that this suboptimal path becomes smoother and have a less travel distance.



(a) Scenario 1



(b) Scenario 2



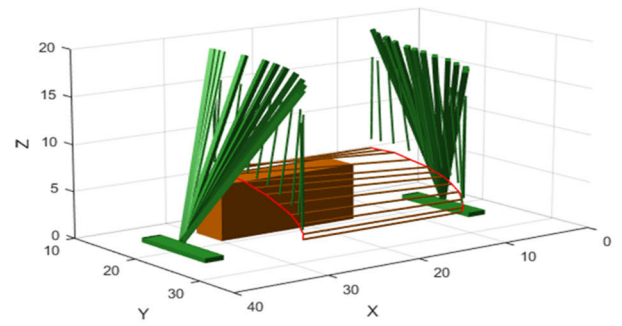
(c) Scenario 3

FIGURE 10. Dual crane simulation for each operation scenario.

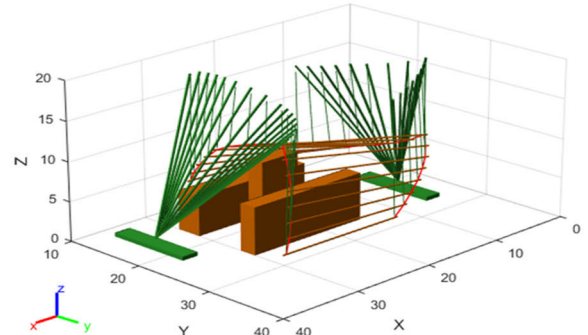
The black dash line and red thick line in Fig. 12(c) show the path before and after applying the smoothing process, respectively. The smoothing process improves the suboptimal raw path ( $Path_{dual}$ ) into more optimal path ( $Path_{dual}^*$ ) that has shorter travel distance in C-space. As shown in Fig. 11(c), the final dual crane cooperative path shows that the girder can be transported smoothly and safely.

Table 2. shows the computation time and number of iterations required to generate a path using the proposed algorithm for each scenario. Each result is an average of 100 path generation results. The time taken to generate a path is fast enough to be used to verify the safety of the operation or to prepare for automated transfer, considering that crane operations normally take minutes to tens of minutes.

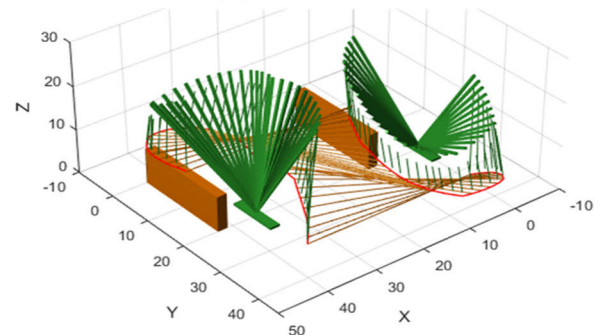
Comparing the computation cost between scenarios, it can be seen that the computation time and iteration increases as the scenario becomes more complex. With a simple



(a) Scenario 1



(b) Scenario 2



(c) Scenario 3

FIGURE 11. Simulation of path generated by the proposed algorithm.

environment like scenario 1, the path is completed in less than 10 iterations, indicating that the proposed algorithm efficiently generates paths in open C-space with a small number of iterations. For Scenario 2, which requires generating a detour route with additional obstacles, the computational cost was about twice that of Scenario 1, but still allowed to generate the path in about 10 iterations.

Scenario 3 requires significantly more time and iterations than Scenarios 1 and 2, indicating that the difficulty of path generation is more challenging.

Most of the increase in computation time comes from the iterations to generate raw paths with multiple RRTs. Since smoothing process repeats with predetermined number of times,  $n_{smooth}$ , there is no significant difference in the time required regardless of the complexity of the scenario. Dividing the time spent on iterations by the number of iterations gives us a time per iteration of 0.98 sec, 1.17 sec, and 1.26 sec

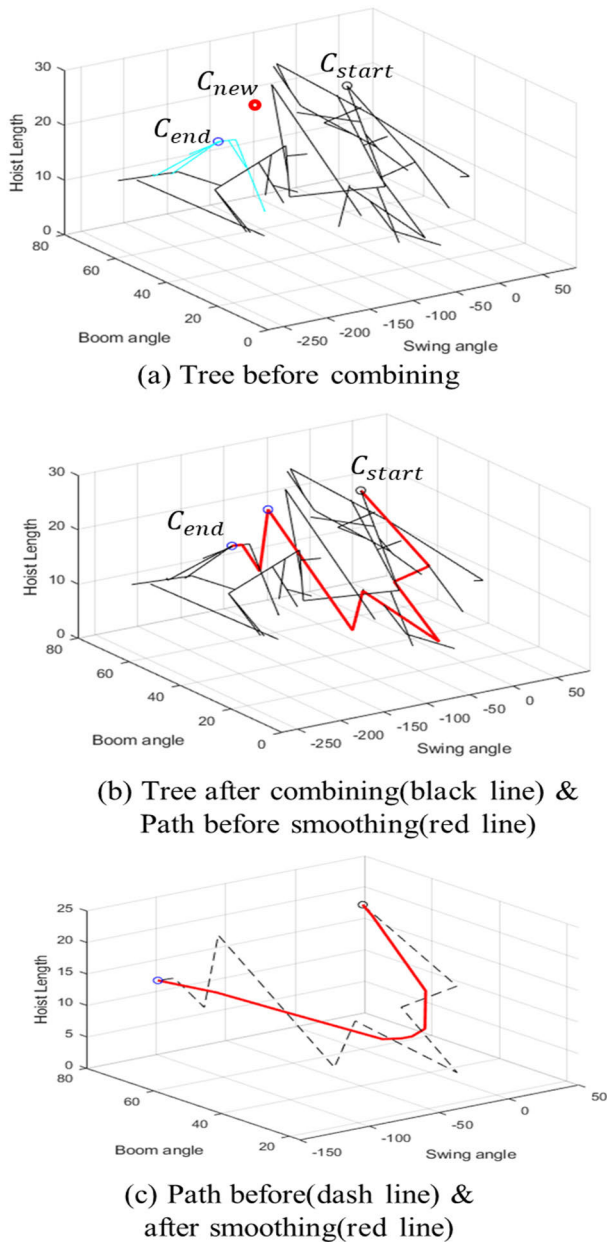


FIGURE 12. Simulation of path generated by the proposed algorithm.

for each scenario. The reason for the increase in time per iteration is that the total number of nodes and trees increases with each iteration, which increases the time required to compute the nearest node and nearest edge of each tree.

In this study, the path generation algorithm was computed in the MATLAB environment, and it is expected that the computation time can be improved by using a lower level language or an environment with higher computing power.

Lastly, we discuss the parameters  $d_{step}$ ,  $n_{smooth}$ , and  $\varphi_{h,max}$  that affect the quality and computation speed of the paths generated by the proposed algorithm.

$d_{step}$  determines how much the girder move in real space when performing a feasibility check for validating an edge

TABLE 2. Computational cost for path generation.

		Scenario 1	Scenario 2	Scenario 3
Average computational time (sec)	Time for iteration	5.22	14.21	53.92
	Time for smoothing	5.95	8.52	8.24
	Total	11.17	22.73	62.16
Average iteration number		5.3	12.1	42.9

TABLE 3. Computational time for different  $\varphi_{h,max}$  value.

$\varphi_{h,max}$	Scenario 1	Scenario 2	Scenario 3
20 degree	10.59	17.47	48.92
10 degree	11.17	22.73	62.16
5 degree	20.53	38.37	150.48

between two configurations. Too small  $d_{step}$  will increase computation time, and too large  $d_{step}$  will miss collisions that may actually occur.

When setting  $d_{step}$ , you need to find and set the largest value that does not miss collisions through simulation. and size information such as boom and girder thickness of the actual crane. When setting  $d_{step}$ , it is necessary to find and set the largest value that does not miss collisions through simulation, and this value can be inferred from thickness of the boom and girder of the actual crane. For example, if the boom is 1 m thick and this is reflected in the simulation, it is not necessary to set  $d_{step}$  to 0.5, 0.1 m or less. If the same type of crane and lifting object are used, once  $d_{step}$  is determined, it does not need to be adjusted afterwards.

$n_{smooth}$  determines the number of iterations of the smoothing process, so adjusting the parameter will change the time required for the smoothing process proportionally. After experiment in various scenarios, it was found that the setting  $n_{smooth}$  to 20 is sufficient to improve the optimality of path. If the generated final path does not have a smooth enough shape in C-space or in real-space simulation, it is possible to further improve the quality of the path by running the smoothing algorithm as many times as desired.

For the third parameter,  $\varphi_{h,max}$ , there are more things to consider when setting an appropriate value compared to the previous two parameters. Depending on the weight of the lifting object and the boom length, the moments can be larger for the same tilting angle of hoist cable, so it is necessary to adjust the value of  $\varphi_{h,max}$  depending on the scenario. Also, as mentioned earlier, in actual operation, swing motion of the suspension subsystem may occur during the transfer, which may cause fluctuations in the moment acting on the crane. Therefore, it is necessary to set the parameter values more conservatively.

Too small value of  $\varphi_{h,max}$  can lead to a rapid increase in computation time for path generation. Table 3 shows the path generation time for different values of  $\varphi_{h,max}$ .

In Scenarios 1 and 2, loosening the constraint on  $\varphi_{h,max}$  from 10 to 20 degrees reduced the path time by about -5% and -11.7% of the path generation time, respectively. When  $\varphi_{h,max}$  decreases from 10 degrees to 5 degrees, the time taken increases significantly, about +84% and +112%, respectively. The main reason for the increase in computation time is that new nodes and new edges candidates should pass a stricter feasibility check during the new node generation and tree connection process. Regardless of the value of  $\varphi_{h,max}$ , the shape of the generated paths for scenario 1 and 2 are not different.

Table 3 shows that  $\varphi_{h,max}$  has a larger impact on computation time for Scenario 3. When the constraints are tightened from 10 degrees to 5 degrees, the time required increases by 142%, and when the constraints are loosened by 20 degrees, the computation time decreases by -21%. As the constraints on the  $\varphi_{h,max}$  became stricter, the generated paths became less smooth, resulting in lower quality suboptimal paths. Since adjusting the  $\varphi_{h,max}$  condition corresponds to increasing or decreasing the size of the door through which a narrow path must be passed, the difficulty and computation time of path generation are sensitive to the setting of  $\varphi_{h,max}$ . If constraint on  $\varphi_{h,max}$  becomes more tightened to limit within 1 degree, the path is difficult to find, even after hundreds of iterations. This is because, under the condition that  $\varphi_{h,max} < 1$  degree, it is very difficult to find the twisting behavior that inevitably occurs to pass through the between cranes. From this result, it can be expected that the path generation algorithms proposed in previous studies that strictly restrict the tilting of the hoist cable to 0 degrees will have difficulty in generating feasible paths for narrow path finding cases such as scenario 3.

## V. CONCLUSION

This study presents a novel multiple rapidly-exploring random trees (RRT) based algorithm specifically designed for dual crane systems, with a focus on addressing the challenges of ‘narrow path finding’ scenarios. The narrow path finding case is defined to represent the case where it is difficult to generate a path in C-space due to avoiding all the constraints, such as when the construction site is too complex with many obstacles or when there are restrictions on the placement of cranes due to space constraints. The objective is an algorithm that efficiently generates appropriate paths in the general case, including narrow path finding. Proposed path planning algorithm not only adapts the multiple RRT method to dual crane systems but also incorporates strategies to enhance path optimality while reducing computational time. The step-by-step implementation of the proposed algorithm has also been described in detail. The effectiveness of our algorithm has been validated through 3 distinct scenarios. Results from these scenarios demonstrate that the proposed algorithm consistently generates smooth and collision-free paths within a reasonable number of iterations. A better understanding of the proposed algorithm was provided by demonstrating the process of tree growth, connectivity, and path smoothing within C-space. The study also provides guidelines for setting

key parameters that have a significant impact on the performance of the algorithm. Further research is needed to improve the computational efficiency of the proposed algorithm with respect to collision detection algorithm and new node generation process. We will also study how to utilize the existing tree to generate adjusted path in real-time when unexpected obstacles are encountered along the path during operation. The proposed research will be further developed in the direction of integration with an automated transfer dual crane system.

## APPENDIX

### SOLVE THE KINEMATIC CHAIN OF SUSPENSION PROBLEM

Given  $x_1, z_1, x_2, z_2, l_{h,1}, l_{h,2}, l_{obj}, \varphi_{h,1}$  in Fig. 2, we will use (3) and (4) to find  $\varphi_{h,2}$ . After organizing the (3) and (4) for  $\cos(\varphi_{obj})$  and  $\sin(\varphi_{obj})$  and squaring and adding two equations to remove the term  $\varphi_{obj}$ , we can obtain new equation for  $\sin(\varphi_{h,2})$  and  $\cos(\varphi_{h,2})$  as shown below.

$$A \sin(\varphi_{h,2}) + B \cos(\varphi_{h,2}) + C = 0 \quad (A1)$$

$$A = 2(x_1 - x_2)l_{h,2} + 2l_{h,1}l_{h,2} \sin(\varphi_{h,1}) \quad (A2)$$

$$B = 2(z_1 - z_2)l_{h,2} - 2l_{h,1}l_{h,2} \cos(\varphi_{h,1}) \quad (A3)$$

$$C = (x_1 - x_2)^2 + (z_1 - z_2)^2 + l_{h,1}^2 + l_{h,2}^2 + 2(x_1 - x_2)l_{h,1} \sin(\varphi_{h,1}) - 2(z_1 - z_2)l_{h,1} \cos(\varphi_{h,1}) \quad (A4)$$

In (A1), after flipping  $B \cos(\varphi_{h,2})$  to the other side, and then squaring both sides, we get the formula for  $\sin(\varphi_{h,2})$  as follow.

$$\left(A^2 - B^2\right) \sin^2(\varphi_{h,2}) + 2AC \sin(\varphi_{h,2}) + \left(C^2 - B^2\right) = 0 \quad (A5)$$

Solving the quadratic equation for  $\sin(\varphi_{h,2})$  with the quadratic formula and applying the  $\sin^{-1}(\cdot)$  function will provide the value of  $\varphi_{h,2}$ . If the result of solving quadratic equation is a complex number or not in the range  $[-1, 1]$ , then there is no valid value of  $\varphi_{h,2}$  for that root.

## REFERENCES

- [1] W. C. Hornaday and C. T. Haas, “Computer-aided planning for heavy lifts,” *J. Construct. Eng. Manage.*, vol. 119, no. 3, pp. 498–515, 1993.
- [2] K.-L. Lin and C. T. Haas, “Multiple heavy lifts optimization,” *J. Construct. Eng. Manage.*, vol. 122, no. 4, pp. 354–362, Dec. 1996.
- [3] H. I. Shapiro, J. P. Shapiro, and L. K. Shapiro, *Cranes and Derricks*. New York, NY, USA: McGraw-Hill, 1991.
- [4] H. R. Reddy and K. Varghese, “Automated path planning for mobile crane lifts,” *Comput.-Aided Civil Infrastruct. Eng.*, vol. 17, no. 6, pp. 439–448, Nov. 2002.
- [5] S. Redon, Y. J. Kim, M. C. Lin, and D. Manocha, “Fast continuous collision detection for articulated models,” in *Proc. 9th ACM Symp. Solid Modeling Appl.*, 2004, pp. 145–156.
- [6] Y.-C. Chang, W.-H. Hung, and S.-C. Kang, “A fast path planning method for single and dual crane erections,” *Autom. Construct.*, vol. 22, pp. 468–480, Mar. 2012.
- [7] J. Olearczyk, A. Bouferguène, M. Al-Hussein, and U. Hermann, “Automating motion trajectory of crane-lifted loads,” *Autom. Construct.*, vol. 45, pp. 178–186, Sep. 2014.

[8] Y. Lin, D. Wu, X. Wang, X. Wang, and S. Gao, "Lift path planning for a nonholonomic crawler crane," *Autom. Construct.*, vol. 44, pp. 12–24, Aug. 2014.

[9] P. Cai, Y. Cai, I. Chandrasekaran, and J. Zheng, "Parallel genetic algorithm based automatic path planning for crane lifting in complex environments," *Autom. Construct.*, vol. 62, pp. 133–147, Feb. 2016.

[10] P. Sivakumar, K. Varghese, and N. R. Babu, "Automated path planning of cooperative crane lifts using heuristic search," *J. Comput. Civil Eng.*, vol. 17, no. 3, pp. 197–207, Jul. 2003.

[11] M. S. A. D. Ali, N. R. Babu, and K. Varghese, "Collision free path planning of cooperative crane manipulators using genetic algorithm," *J. Comput. Civil Eng.*, vol. 19, no. 2, pp. 182–193, Apr. 2005.

[12] P. Cai, I. Chandrasekaran, J. Zheng, and Y. Cai, "Automatic path planning for dual-crane lifting in complex environments using a prioritized multi-objective PGA," *IEEE Trans. Ind. Informat.*, vol. 14, no. 3, pp. 829–845, Mar. 2018.

[13] H. Choset, K. Lynch, S. Hutchinson, G. Kantor, W. Burgard, L. Kavraki, and S. Thrun, *Principles of Robot Motion: Theory, Algorithms, and Implementations*. Cambridge, MA, USA: MIT Press, 2005.

[14] L. E. Kavraki, P. Svestka, J.-C. Latombe, and M. H. Overmars, "Probabilistic roadmaps for path planning in high-dimensional configuration spaces," *IEEE Trans. Robot. Autom.*, vol. 12, no. 4, pp. 566–580, May 1996.

[15] D. Brandt, "Comparison of a and RRT-connect motion planning techniques for self-reconfiguration planning," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, Beijing, China, Oct. 2006, pp. 892–897.

[16] J. R. Bruce, "Robot motion planning," Dept. Comput. Sci., Carnegie Mellon Univ., Pittsburgh, PA, USA, Spring 2004. [Online]. Available: <https://www4.cs.pitt.edu/~jacky/Robotics/Papers/BruceERRTMotionPlanningSlides.pdf>

[17] S. M. Lavalle, "Rapidly-exploring random trees: A new tool for path planning," Dept. Comput. Sci., Iowa State Univ., IA, USA, Rep. 9811, 1998.

[18] W. Wang, Y. Li, X. Xu, and S. X. Yang, "An adaptive roadmap guided multi-RRTs strategy for single query path planning," in *Proc. IEEE Int. Conf. Robot. Autom.*, May 2010, pp. 2871–2876.

[19] M. Clifton, G. Paul, N. Kwok, D. Liu, and D.-L. Wang, "Evaluating performance of multiple RRTs," in *Proc. IEEE/ASME Int. Conf. Mechatronic Embedded Syst. Appl.*, Oct. 2008, pp. 564–569.

[20] S. Karaman and E. Frazzoli, "Sampling-based algorithms for optimal motion planning," *Int. J. Robot. Res.*, vol. 30, no. 7, pp. 846–894, Jun. 2011.

[21] S. Karaman and E. Frazzoli, "Incremental sampling-based algorithms for optimal motion planning," *Robot. Sci. Syst. VI*, vol. 104, no. 2, pp. 267–274, 2010.

[22] Y. Zhou, E. Zhang, H. Guo, Y. Fang, and H. Li, "Lifting path planning of mobile cranes based on an improved RRT algorithm," *Adv. Eng. Informat.*, vol. 50, Oct. 2021, Art. no. 101376.



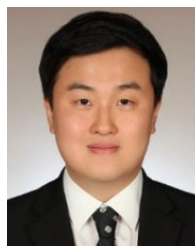
**JI-CHUL KIM** received the bachelor's degree in mechanical engineering from Yonsei University, in 2007, and the master's and Ph.D. degrees in mechanical engineering from KAIST, with a focus on robotics, in 2009 and 2014, respectively. He is currently a Senior Researcher with the Department of Smart Industrial Machine Technologies, Korea Institute of Machinery and Materials (KIMM). His main research interests include unmanned system technology and automatic control.



**HANMIN LEE** received the bachelor's, master's, and Ph.D. degrees in mechanical engineering from KAIST, in 1998, 2000, and 2005, respectively. He is currently a Principal Researcher with the Department of Smart Industrial Machine Technologies, Korea Institute of Machinery and Materials (KIMM). His main research interests include robotics and AI.



**YEONGJAE KIM** received the bachelor's degree in mechanical engineering from Chungnam National University. He is currently an Engineer with the Department of Smart Industrial Machine Technologies, Korea Institute of Machinery and Materials (KIMM). His main research interest includes robotics.



**DONGWOOK LEE** received the B.S., M.S., and Ph.D. degrees in mechanical engineering from Korea Advanced Institute of Science and Technology (KAIST), Daejeon, South Korea, in 2012, 2014, and 2018, respectively. Since 2022, he has been an Assistant Professor with the Department of Mechanical and Automotive Engineering, Kongju National University. His research interests include autonomous system control, vehicle dynamics and control, and motion control.

...