University of Alberta

A Robotic Approach to the Analysis of Obstacle Avoidance in Crane Lift Path Planning

by

Zhen Lei

A thesis submitted to the Faculty of Graduate Studies and Research
in partial fulfillment of the requirements for the degree of

Master of Science
in
Construction Engineering and Management

Department of Civil and Environmental Engineering

Spring 2011
Edmonton, Alberta

EXAMINING COMMITTEE

Committee Chair:

Dr. Ming Lu, Department of Civil and Environmental Engineering, University of Alberta

Co-Supervisors:

Dr. Mohamed Al-Hussein, Department of Civil and Environmental Engineering, University of Alberta

Dr. Saeed Behzadipour, Department of Mechanical Engineering, University of Alberta

Other Examining Committee:

Dr. Michael Lipsett, Department of Mechanical Engineering, University of Alberta

ABSTRACT

Crane lift path planning is time-consuming, prone to errors, and requires the practitioners to have exceptional visualization abilities, in particular, as the construction site is congested and dynamically changing. This research presents a methodology based on robotics motion planning to numerically solve the crane path planning problem. The proposed methodology integrates a database in order to automatically conduct 2D path planning for a crane lift operation, and accounts for the rotation of the lifted object during its movements. The proposed methodology has been implemented into a computer module, which provides a user-friendly interface to aid the practitioners to perform a collision-free path planning, and check the feasibility of the path at different stages of the project. Three examples are described in order to demonstrate the effectiveness of the proposed methodology and illustrate the essential features of the developed module.

ACKNOWLEDGEMENT

I would like to express my gratitude to my co-supervisors, Dr. Mohamed Al-Hussein and Dr. Saeed Behzadipour, for their patient guidance and continuous support during this research. I would like to extend my gratitude to my thesis committee, Dr. Ming Lu and Dr. Michael Lipsett, for their suggestions and recommendations.

The support from industrial partner, PCL Industrial Constructors Inc., is gratefully acknowledged. I would like to acknowledge Mr. U. H. Hermann, Mr. Ali Hendi, Mr. Jacek Olearczyk, and Mr. Dony Alex, who has encouraged and supported me in all stages of this research. I am also grateful to Ms. Joanna Klimowicz who patiently improved my writing.

Finally, I wish to give my deepest gratitude to my family, who has supported me all the time.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1: INTRODUCTION

1.1 Research Motivation

The efficiency of construction has been improved dramatically in recent decades due to the use of cranes. Efficient cranes utilization, prefabrication and on-site installation became possible, resulting in an improvement of the productivity and quality, and savings of costs and time. Currently, cranes are widely used, and the mobile crane especially has dominated the North America construction sites (Shapira et al. 2007). Errors during heavy lift planning lead to extra costs and schedule delay. The heavy lift planning is carried out in the pre-construction phase with limited data available. The 2D CAD format (plot plans) are usually provided along with a list of equipment for their weights and dimensions. Based on these plot plans, a preliminary lift study is developed, which contains information such as the crane configuration, the capacity of the crane, load information, as well as a 2D elevation view where the clearances between the lifted object and the obstructions are checked. The lift study aims to answer whether and how the load can be lifted on site.

One task of the lift study is heavy lift path planning, which tries to find a collision-free trajectory for the lifted object among on-site obstacles from its pick location toward final location. Current methods for heavy lift path planning are manual and time-consuming (Shapiro et al. 1999) and lack reaction to the dynamic changes to the construction site, due to newly installed objects.

Industrial construction projects in particular involve frequent heavy lift operations using mobile cranes (Figure 1.1). The need for developing a decision support system, to automate the crane path planning process, is quite eminent. This research presents a robotic motion planning methodology to solve the crane path planning problem in computer environment. The designed system integrates a database and utilizes Visual Basic.Net programming environment in order to develop a prototype module, providing a user-friendly interface to aid the practitioners during the path planning and collision checking. Compared with existing tool CAD which is applied to check of the collision free path, the designed system achieves the automation of the path planning process, and determines the feasible solution in a more efficient way.

1.2 Research Objectives

This research aims to develop a decision support system to automate the process of heavy lift path planning. The main objectives of this research are summarized below:

- To obtain understanding of robotic motion planning applications and crane heavy lift planning;
- To explore robotic motion planning implementation in the construction field;
- To build an algorithm and implement it as a computer program to automate the heavy lift path planning process;
- To assist engineers and practitioners to avoid potential crane lifting accidents, and reduce the time and cost associated with path planning on construction sites.

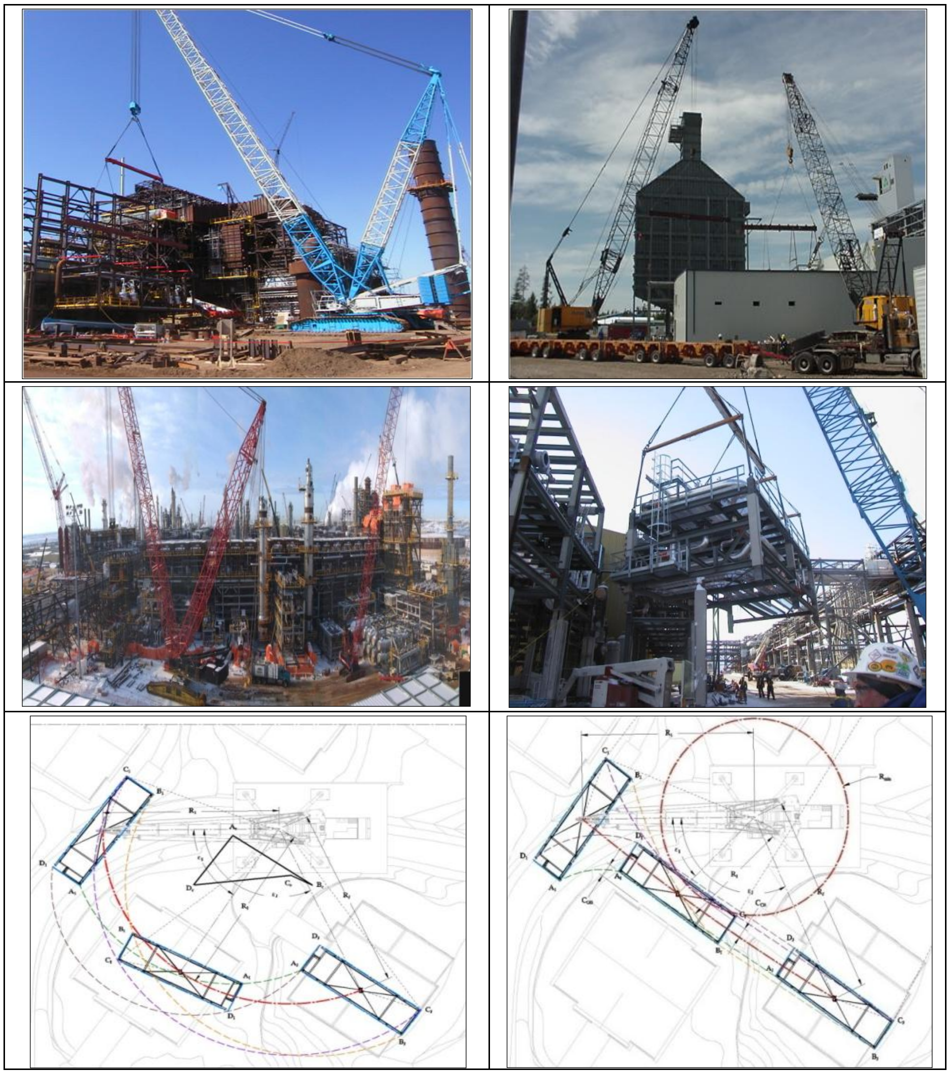


Figure 1.1 Sample Heavy Lifts and Path Planning

1.3 Thesis Organization

Chapter 2 (Literature Review) provides a summary of construction crane heavy lift planning and robotic motion planning. In addition, the applications of both heavy lift planning and the applications of robotic motion planning in construction are introduced.

Chapter 3 (Proposed Methodology) discusses the proposed methodology used in this research. First, the philosophy of motion planning in robotics is described. Then the detailed methods for motion planning are presented.

Chapter 4 (Implementation Process) discusses the development of the path planning system, which includes the detailed algorithms and processes used in this research. Three case studies are provided to represent the effectiveness of the proposed methodology.

Chapter 5 (Conclusions) describes the general conclusions, main contributions, and some recommendations for practical applications and future research.

CHAPTER 2: LITERATURE REVIEW

2.1 Introduction

In this chapter, the applications in the following areas will be presented:

- Heavy Lift Planning Applications for Construction Cranes;
- 4D CAD Applications in Construction
- Automation in Construction;
- Motion Planning in Construction.

2.2 Heavy Lift Planning Applications for Construction Cranes

Heavy lift planning maintains a central role in the success of crane operations, and errors in this process can result in extra cost and schedule delay. Eight criteria are identified as essential to judge the feasibility of a heavy lift planning (Varghese 1997): 1) Availability of crane; 2) Access to site; 3) Access to lift area; 4) Location to execute lift; 5) Lift path clearances; 6) Capacity during lift; 7) Ground support during lift; 8) and removal from lift area. Although it is possible that other criteria could be present, these eight criteria cover the majority of issues related to heavy lift planning.

Several efforts have been made to improve the heavy lift planning process. Many have focused on crane selection; for instance, Al-Hussein (1999) proposed a methodology for crane selection location and on-site utilization for construction projects, and also an optimization algorithm for selection and location of mobile cranes on construction sites (Al-Hussein et al. 2005). A fuzzy logic approach was applied to crane type selection by Hanna and Lotfallah (1999). Also, a crane selection tool, IntelliCranes, has been developed based on probabilistic neural networks by Sawhney and Mund (2002). Tam et al. (2001) and Tam and Tong (2003) applied a genetic algorithm (GA) to analyze the operations of tower cranes. In addition, Manrique et al. (2007) described a methodology used to integrate crane selection algorithms and an optimization model with 3D modeling and animation for the selection, utilization, and location of cranes. An optimization model has also been developed by Zhang et al. (1999) to optimize the location of a group of tower cranes. A database management system was developed to assist practitioners to manage crane-related data by Al-Hussein et al. (2000) to house information related to cranes, their geometric lifting configuration specifications, and their lifting capacities based on the information provided by manufacturers in crane lifting capacity charts. Hasan et al. (2010) presented a newly automated system for preparing lift studies and designs for a mobile crane supporting system.

2.3 4D CAD Applications in Construction

4D CAD integrates the 3D graphical model with the construction schedule, which assists the practitioners to visualize the construction processes in a 3D environment. An evaluation by Mahalingam, A., et al. (2009) has shown that 4D CAD can deliver benefits in construction management. The concept of 4D CAD was first introduced by the Center for Integrated Facility Engineering (CIFE) at Stanford University. Compared with 3D CAD, 4D CAD imports time information into a static 3D model, which is able to display a construction schedule in a 3D environment. In Stanford, Collier and Fischer (1995) applied 4D CAD technology to a construction project. After that, a 4D CAD tool, CIFE 4D-CAD, was developed to generate the 4D model within one single environment (Mckinney et al. 1996). Wang, et al. (2004) developed a 4D Site Management Model+ (4DSMM+) for addressing the need for linking scheduling data to a 3D computer graphics building model. Ma et al. (2005) proposed a 4D CAD Integrated Site Planning System (4D-ISPS) system, which integrates schedules, 3D models, resources and site spaces to provide 4D graphical visualization capability for construction site planning.

2.4 Automation in Construction

A survey pointed out that Japan has been a leading force in the implementation of automation in construction field, in terms of the number of systems under use or development, followed by US, Germany and UK (Warszawski and Navon 1998). In North America, the concept of applying robotic technology in construction emerged since the early 1900s (Kangari and Halpin 1989; Warszawski 1990a and 1998; Everett and Slocum 1994). Warszawski 1990b presented the classifications of robots for different uses in construction: 1) Exterior handling robots; 2) Horizontal finishers; 3) Vertical finishers; 4) Interior finishers. Kangari and Halpin (1989) pointed out the processes with best opportunities for robotic applications in construction: 1) Steel fabrication; 2) Painting; 3) Wall finishing; 4) Bush hammering; 5) Tunneling; 6) Sandblasting; 7) Concrete placement; and 8) Fireproof spraying. It was anticipated that automation application could result in approximately 10-15% increase in overall construction productivity rate (Skibniewski and Russell 1989). Robots for different use have been developed for the construction field (Warszawski et al. 1990). Besides that, some robotic methodologies are also applied into construction field, and robotic motion planning is one of those.

2.5 Robotics Motion Planning Applications in Construction

The ultimate goal for robotics is to create autonomous robots, which can accept high-level descriptions of tasks and will execute these tasks without human intervention. (Latombe 1991). To achieve this, several problems should be addressed, one of which is motion planning. Motion planning aims to ensure collision avoidance for the moving robot(s). It also guarantees that the movements of the robot(s) are efficient, with respectively short moving distance and without unnecessary movements. There are many types of motion planning problem; for example, the robots may travel with obstacles moving at the same time. The problem faced in this research is a basic motion planning problem, which satisfies two assumptions: 1) Only one robot in the work space (see next paragraph for the definition of work space); 2) The locations of the obstacles are fixed.

Terminologies which are widely used for motion planning problem are introduced (the stated terminologies are quoted from a survey by Hwang and Ahuja 1992). In robotics, the robots refer to the things that are moving, whether points, polytopes[1], or manipulators[2]. When the robots move, they are constrained by their surroundings. The physical space in which robots and obstacles exist is called a work space. To represent the position of a robot of a given shape, a configuration is proposed. It contains a set of independent parameters that characterize the position of every point in the robot. These independent parameters are called degree of freedom (DOF). The set of all configurations is called configuration space. Configurations that result in collisions between the robot and obstacles are called configuration

[1] Polytopes are polygons in 2D or polyhedral in 3D.

[2] A manipulator is a mechanical arm consisting of links and joints.

obstacle (C-Obstacle). The free space refers to parts of the configuration space for which the robot does not collide with any obstacle. In general, the motion planning problem is consists of two steps: 1) convert the work space to a configuration space; and 2) search for the collision-free path for the robot from its starting configuration to ending configuration through the configuration space. These two parts will be introduced in section 2.3.1 and 2.3.2 respectively.

2.5.1 Configuration Space Approach

To solve a motion planning problem, the first step is to convert a work space to a configuration space, the method of which is called configuration space approach. It has been applied to the field of kinematics, with many applications including robotics path planning, packing and nesting, automated assembly, etc. The configuration space approach was initially proposed by Lozano-Pérez (1983), as converting the world space into configuration space by shrinking the object into a representing point. The obstacles in work space are also converted into C-Obstacles (this method will be detailed in section 3.2.1).

Lozano-Pérez's (1983) proposed a method called obstacle growth to generate the configuration space by using Minkowski point-set operations, which is widely used in the motion planning problem. The core idea is to reduce the shape of the robot to a representing point, create the C-Obstacles, and search the path for this representing point. This method simplifies the path planning problem, avoiding checking all the possible collisions between the entire shape of the robot and the obstacles. This method was further developed by other researchers (Like Gouzènes 1984). The creation of C-Obstacles considers the degree of freedom (DOF) of the robot. It has been stated that the configuration space approach could become complicated

(Hwang and Ahuja 1992). For example, for a rigid robot in 3D environment, it needs six degree of freedom (DOF) to define the configuration of the robot, so consequently the C-space will be 6-dimensional. In this case, representing the C-space with a grid requires 10^{12} points for a resolution of 100 points per dimension, which is challenging.

2.5.2 Motion Planning Approaches

Motion planning approaches are implemented in order to find the path for the robot from the initial configuration to the goal configuration. Latombe (1991) reviewed the classic work of motion planning approaches. The common methods are roadmap, cell decomposition, and potential field.

The Roadmap method aims to capture the connectivity of the robot's free space in a network of one-dimensional curves. Two typical types of roadmaps exist: Visibility graph (Asano 1985), and Voronoi diagram (Aurenhammer 1991). A visibility graph is composed of nodes, the initial and goal configurations of the robot, and all the C-Obstacle vertices. The feasible path for the robot is searched through the connections among these nodes. A voronoi diagram creates a buffer around the C-Obstacles, and yields the path for the robot outside of the buffers. This method increases the clearance between the robot and the obstacles.

Cell decomposition decomposes the free space into small regions, called cells, then connects the adjacent cells, and at last the path planning can be conducted between adjacent cells rather than individual points. Cell decomposition can be categorized into the exact cell decomposition and approximate cell decomposition methods (Latombe 1991).

Potential field proposes that the goal configuration can generate an “attractive force”, which drags the moving robot toward it, while the C-Obstacles produce a repulsive force which forces the moving robot away from the C-Obstacles. Both “attractive force” and the “repulsive force” affect the moving robot at the same time and consequently determine its direction of motion.

2.5.3 Motion Planning Applications in Construction Cranes

One field of application for automation in construction is to apply robotic motion planning methodology to solve construction problems. Two examples of achievements of robotic motion planning applications in construction are the creation of a computer-aided construction system for a shotcreting[3] robot (Cheng et al. 2001), and robot path-planning for earthwork operations in construction (Kim et al. 2003). Also, robotic motion planning has been implemented to handle crane lift path planning problem. Researchers like Ali et al. (2005) and Sivakumar et al. (2003) solved cooperative crane lifts path problem by using motion planning method. Different search path methods have been used (Ali et al. 2005 used a genetic algorithm (GA); Sivakumar et al. 2003 applied a heuristic search). Reddy et al. (2002) developed a system for automated path planning for single mobile crane lifts using the AutoCAD environment and AutoLisp. A virtual crane model has been built for visualization of erection processes and erection schedule by using robotic motion planning (Kang and Miranda 2006). To our knowledge, most of the previous works considered the crane and the lifted object as a manipulator. Based on its degree of freedom (DOF), the corresponding configuration space is generated. For each

[3] Shotcrete is a construction technique, which conveys the concrete through a host and projects it at high velocity onto a surface.

configuration of the manipulator, the interference detection is conducted to check whether the configuration collides with the obstacles. Eventually by implementing various search methods, the path is obtained. The algorithm introduced in this thesis views the lifted object as the manipulator, and generates the C-Obstacles. Instead of detecting the interference among all configurations, the path is searched through the free space.

CHAPTER 3: PROPOSED METHODOLOGY

3.1 Overview

This research is based on the approach as shown in Figure 3.1. A set of inputs (for example, capacity of the crane, coordinates of the obstacles, etc.) are considered. The main process combines robotic methodology with current construction practice and develops an algorithm, which is implemented in a computer environment. The outputs of this research are feasible path for crane operations, path sensitivity analysis, and modified path, which are subjects to the criteria (for example, the schedule of the crane lift, safety specifications, etc.). Also, two general assumptions are made for this research: 1) The minimum and maximum radius of the crane and boom clearance are represented as virtual obstacles in the moving space; and 2) Since most on-site obstacles of industrial construction projects are usually in rectangle shape, the system is designed to handle the obstacles which are convex shape. The concave obstacles can be divided into convex obstacles manually and input into the database as separate ones.

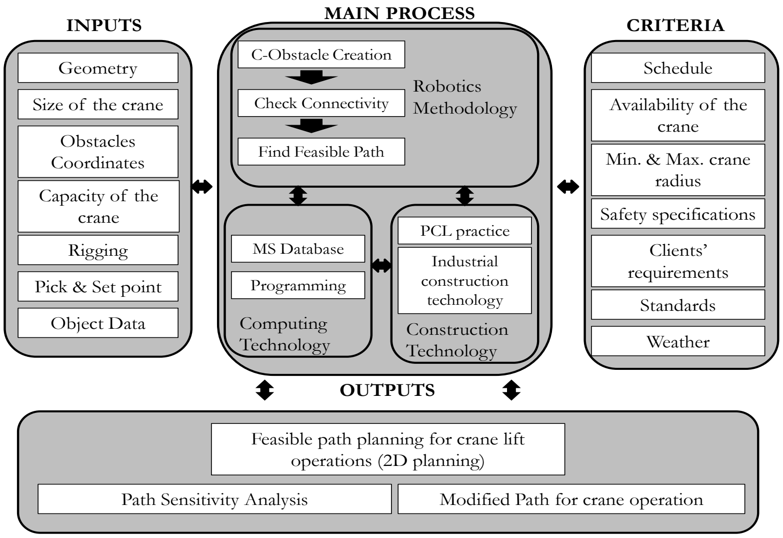


Figure 3.1 Overview of the Research Approach

The main idea of the robotic methodology adopted in this research is to convert the lifted object to a representing point, and then corresponding C-Obstacles are generated. This method simplifies the path planning problem to searching the collision-free path for the representing point, meanwhile avoiding checking the clearance of the lifted object for its entire shape. This robotic methodology consists of three steps as stated below.

In step one, the C-Obstacles are generated by the method called obstacle growth, which will be detailed in Section 3.2 (Lozano-Pérez 1983). C-Obstacles are generated based on the real obstacles and by the shape of lifted object, and the entire lifted object is represented as a representing point. No collision exists between the lifted object and the obstacles if the representing point travels outside or on the edges of C-Obstacles, otherwise, collision occurs. In this case, finding the path for the lifted object is equivalent to searching the path for the representing point through the C-Obstacles.

The shapes of the C-Obstacles are changed if the lifted object is rotated during its movements. Therefore, C-Obstacles corresponding to different rotation angles of the lifted object should be created. The C-Obstacles corresponding to the same rotation angle of the lifted object are put on one layer. Then different layers are generated for different rotations. On each layer, the C-Obstacles are represented by the coordinates of their vertices, also called nodes. After all the layers are generated, the next step is to check the connections of the nodes on the same layer and between successive layers.

Step two focuses on finding the connections between nodes on the same layer and between two successive layers. The method of checking the connections on the same layer is called a visibility graph (Asano 1985), which will be detailed in section 3.3. All connections, represented by their lengths, will be stored in an adjacency matrix. In step three, the Dijkstra's algorithm is applied to search the shortest path based on the generated adjacency matrix.

A central database is constructed to support all the calculations, which has the six entities (tables) ("Obstacles table", "Pick table", "Place point table", "Convex hull table", "Merged convex hull table", and "Solution table"). These entities (tables) interact, following the sequence illustrated in Figure 3.2, which includes four steps: 1) the user inputs the coordinates of obstacles, pick point and end point into the first three tables ("Obstacle tables", "Pick point tables", and "Place point tables"); 2) the system automatically read the data and generate the C-Obstacles, and records the data of the C-Obstacles in the "Convex hull table"; 3) the program uses a library to modify the generated C-Obstacles, and the information in the merged "Convex hull database" is used to check the connectivity and consequently find the feasible path; and 4) the final solution is presented in a 2D graphical interface.

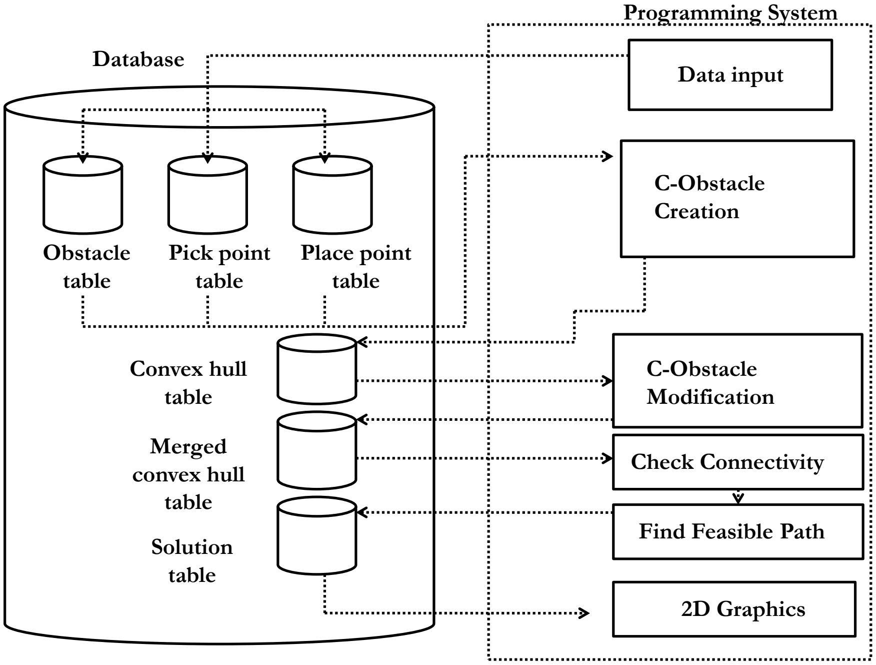


Figure 3.2 Database and Programming System

3.2 Configuration Space Obstacle Creation

3.2.1 Overview

The basic idea of creating C-Obstacles is to reduce the lifted object into a representing point, and grow the obstacles by the shape of the lifting object. For example, as shown in Figure 3.3, there is a triangular lifted object *A*, and a rectangular obstacle *B*. To generate the C-Obstacle for *B*, the first step is to choose one vertice as the representing point (in this case, the top vertice of the lifted object is chosen as the representing point). The second step is to grow obstacle *B* by the shape of *A*. Consequently, the problem of finding a path for *A* relative to *B* is equivalent to finding a path for the representing point relative to C-Obstacle. When the representing point moves outside the boundary of the C-Obstacles, no collision exists between the lifted object and the obstacle; if it is on the edge of the C-Obstacle, the lifted object is just touching the obstacle. However, the movements of the representing point in the C-Obstacle results in a collision.

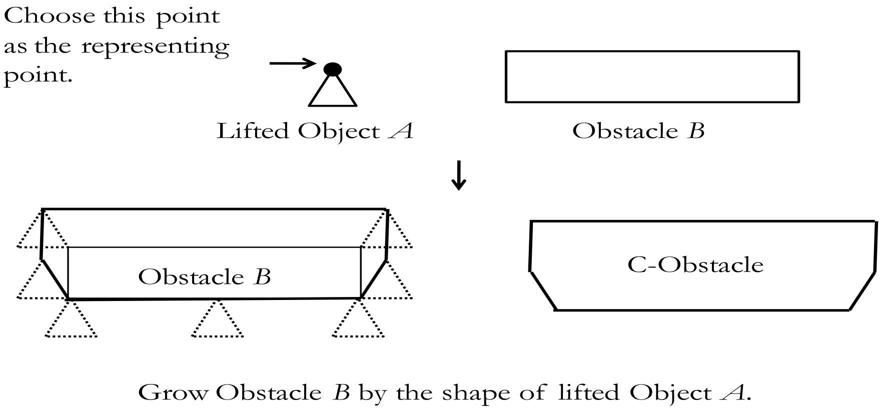

Choose this point as the representing point.
Lifted Object *A*
Obstacle *B*
Obstacle *B*
C-Obstacle
Grow Obstacle *B* by the shape of lifted Object *A*.

Figure 3.3 Obstacle Growth Approach

In practice, the rotations of the lifted object should be considered. The lifted object can be rotated at certain locations on a construction site, and accordingly the shapes of the generated C-Obstacles vary as the change of the rotation of the lifted object. Ideally, the method described in Figure 3.3 needs to be carried out for every possible rotation of lifted object to create C-Obstacles. However, to simplify the problem, C-Obstacles are created based on a discrete rotation step, which means the lifted object is rotated within a defined degree discretely every time. For example, as illustrated in Figure 3.4, a discrete rotation step is defined as 120° (In real practice, the rotation step should be smaller). Then instead of considering all possible rotations from 0° to 360°, only 3 rotations (360°/120°=3) are handled and corresponding C-obstacles are obtained. Each set of C-obstacles are considered as one layer and totally three layers are generated, namely 0°, 120°, and 240° (Figure 3.4). If the discrete rotation step is less than 120°, more layers are generated since the value of (360°/discrete rotation step) became smaller.

A group of layers can be generated with the defined rotation step. On each layer, the generated C-Obstacles are represented by the coordinates of their vertices in either a clockwise or counter-clockwise order. As shown in Figure 3.5, different layers represent the C-Obstacles corresponding to different rotation angles (in addition to the "X" and "Y" dimensions, "rotation" is the third dimension). Meanwhile, nodes that exist on one of these layers represent the shapes of the C-Obstacles for one specific rotation of the lifted object. After generating all layers, the next step is to check the connections of the nodes on each layer and from successive layers. Section 3.3.2 will detail the algorithm mentioned in this section before proceeding to the algorithm of checking the connections.

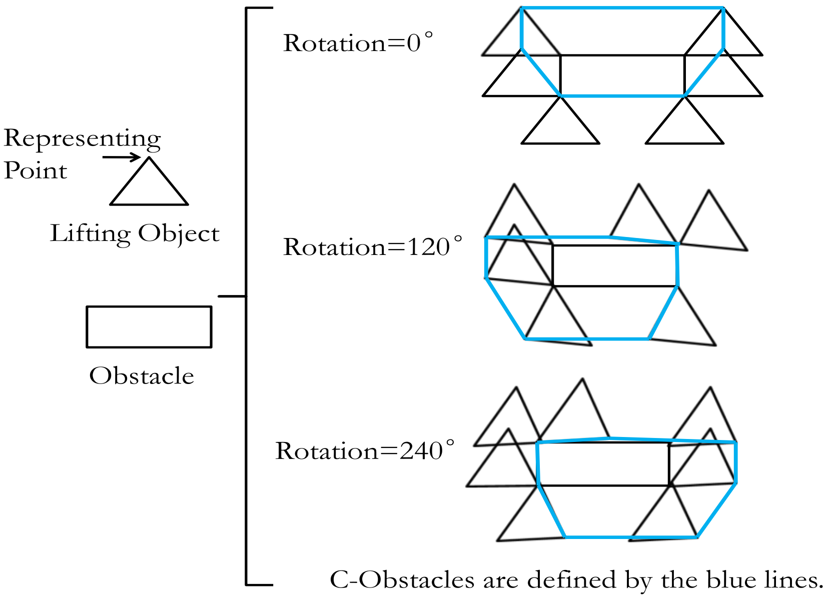


Figure 3.4 One Example of Creating C-Obstacles for Different Rotations

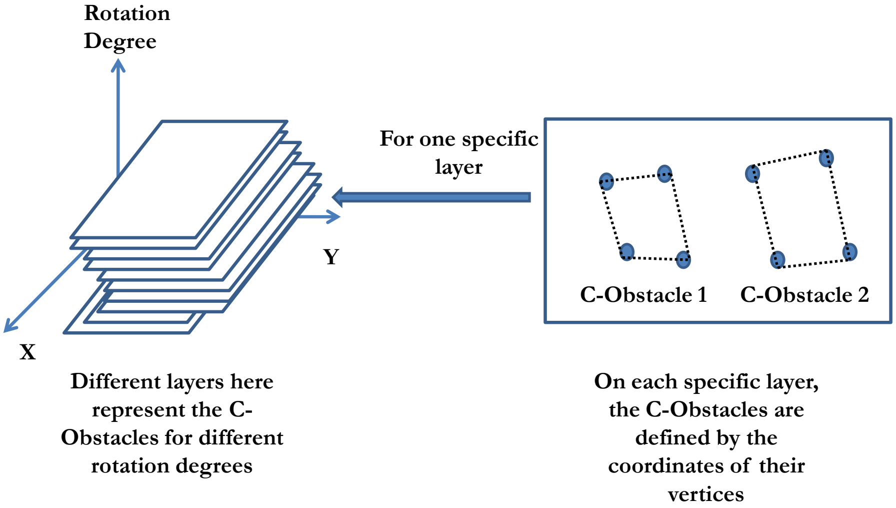

Rotation Degree
Y
X
For one specific layer
C-Obstacle 1
C-Obstacle 2
Different layers here represent the C-Obstacles for different rotation degrees
On each specific layer, the C-Obstacles are defined by the coordinates of their vertices

Figure 3.5 Combinations of Layers

3.2.2 The Algorithms for Creating C-Obstacles

The algorithm for generating the C-Obstacles contains four main steps: (1) Create vectors of the lifted object; (2) Add vectors to the obstacles; (3) Generate convex hull (C-Obstacles); and (4) Loop rotation. The flowchart of the algorithm is given in Figure 3.6. The four steps will be discussed in detail as follows (Step#1, Step#2, Step#3, and Step#4).

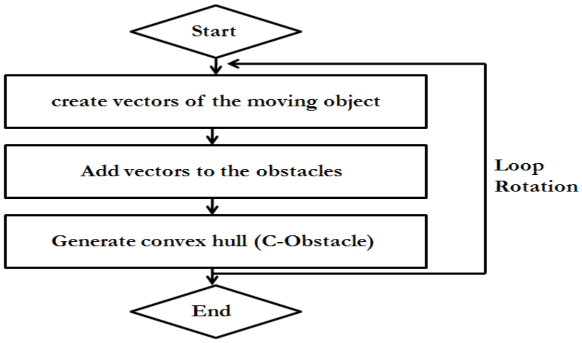

Start
create vectors of the moving object
Add vectors to the obstacles
Generate convex hull (C-Obstacle)
Loop Rotation
End

Figure 3.6 Flowchart of creating C-Obstacles

Step# 1 Create vectors of the lifted object: The first step is to create vectors based on the representing point of the lifted object. As shown in Figure 3.7, "part 1" illustrates a lifted object with 4 nodes (numbered in a clockwise order). Node 1 is chosen as the representing point (The representing point can be any point on the lifted object), and three vectors are created from the other nodes of this lifted object towards node 1. Generally, for any shape of lifted object, it contains *n* vertices defined by(x_i^{Obj}, y_i^{Obj}) $(i = 0,1,2,3, \ldots, n)$. The representing point is chosen and defined as $(x_0^{Obj},\ y_0^{Obj})$. Then the vectors, which point from the nodes other than the representing point (x_i^{Obj} $(i = 1,2,3, \ldots, n)$, y_j^{Obj} $(j = 1,2,3, \ldots, n)$) toward the representing point, are obtained and stated as $\vec{x}i$ and $\vec{y}i$ $(i = 1,2,3, \ldots, n)$, calculated satisfying Equations (1) and (2) respectively:

$$\vec{x}i = x_0^{Obj} - x_i^{Obj}(i = 1,2,3, \ldots, n) \tag{1}$$

$$\vec{y}i = y_0^{Obj} - y_i^{Obj}(i = 1,2,3, \ldots, n) \tag{2}$$

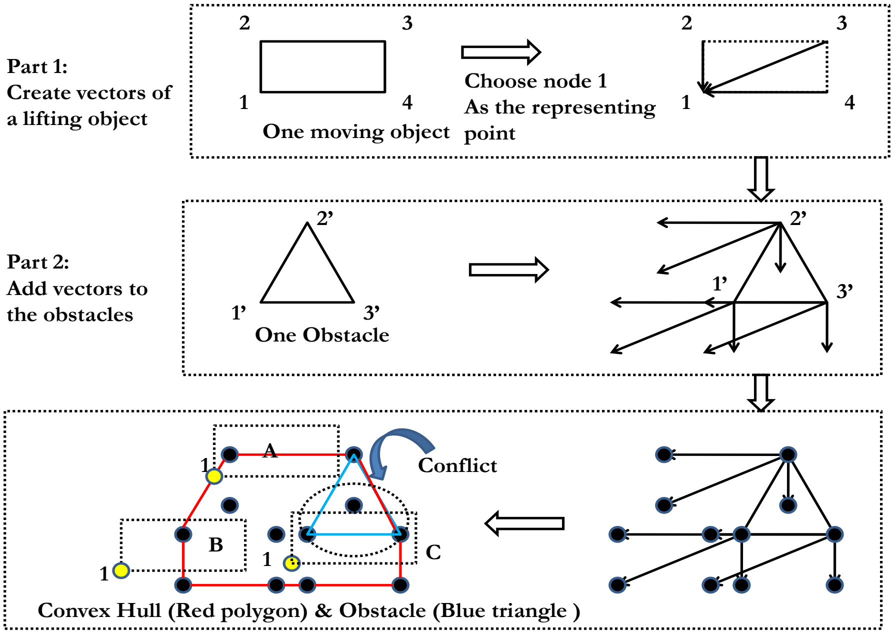


Figure 3.7 Step# 1 (Create vectors) & Step# 2 (Add vectors)

Step# 2 Add vectors to the obstacles: As shown in "Part 2" of Figure 3.7, an triangular obstacle has three nodes, 1', 2', and 3'. The vectors generated from "Part 1", are added to each node of the obstacle. After adding, each node is extended into 4 nodes (includes the original node itself), so that there are 12 nodes in total. Let us generally assume there is one obstacle, and it has *m* vertices. So the obstacle can be presented by its vertices, which are defined as (x_i , y_i $(i = 0,1,2,3, \dots, m)$). Also, it is assumed that there are *n* vectors generated from "Step# 1", denoted by $(x_j, y_j\ (j = 1,2,3, \dots, n))$. Therefore, new nodes for this obstacle after adding the vectors are calculated by Equations (3) and (4) respectively.

$$x_i^j = x_i + x_j(i = 1,2,3, \dots, m; j = 0,1,2,3, \dots, n) \quad (3)$$

$$y_i^j = y_i + y_j(i = 1,2,3, \dots, m; j = 0,1,2,3, \dots, n) \quad (4)$$

Step# 3 Generate convex hull: The addition of vectors to the obstacles generates a group of nodes (as shown in Figure 3.7, "Part 2"). The next step is to generate the convex hull (C-Obstacle), which embraces all the nodes generated from "Step# 2". as shown by the red polygon in Figure 3.7, the convex hull contains all the nodes from "Part 1" within or on its edges. Meanwhile, the obstacle is shown as the light blue triangle. Since node 1 has been chosen as the representing point, so if it travels outside or on the edge of the convex hull, as the yellow nodes of cases "A" and "B" in Figure 3.7, there is no collision between the lifted object and the obstacle. On the other hand, if node 1 enters the convex hull, as the yellow node of case "C" in Figure 3.7, collision occurs. The process of generating the convex hull is discussed as below.

In Figure 3.8 "Procedure 1", we assume that there are many generated nodes, and intend to find a boundary (convex hull) to embrace all these nodes. There are 4 sub-steps to generate the convex hull. First, the leftmost node is detected among all the existed nodes with the minimum x coordinate (Node *A* in "Procedure 1" in Figure 3.8). Node *A* is denoted by (x_A, y_A). Then a positive real number *d* is deducted from x_A, and node *B* is obtained by $(x_A - d, y_A)$, shown as the red node in "Procedure 1" in Figure 3.8. A horizontal line segment is created by linking nodes *A* and *B*. Then Node *A* is connected to nodes on its right side sequentially, and for each connection, the connection line segment is denoted by *AC*. The different clockwise angles *BAC*s are obtained, and smallest one can be found. The node which gives the smallest *BAC* angle is marked as one of the vertices of the convex hull. Also, node *A* in "Procedure 1" is noted as a node on the edge of the convex hull (C-Obstacle).

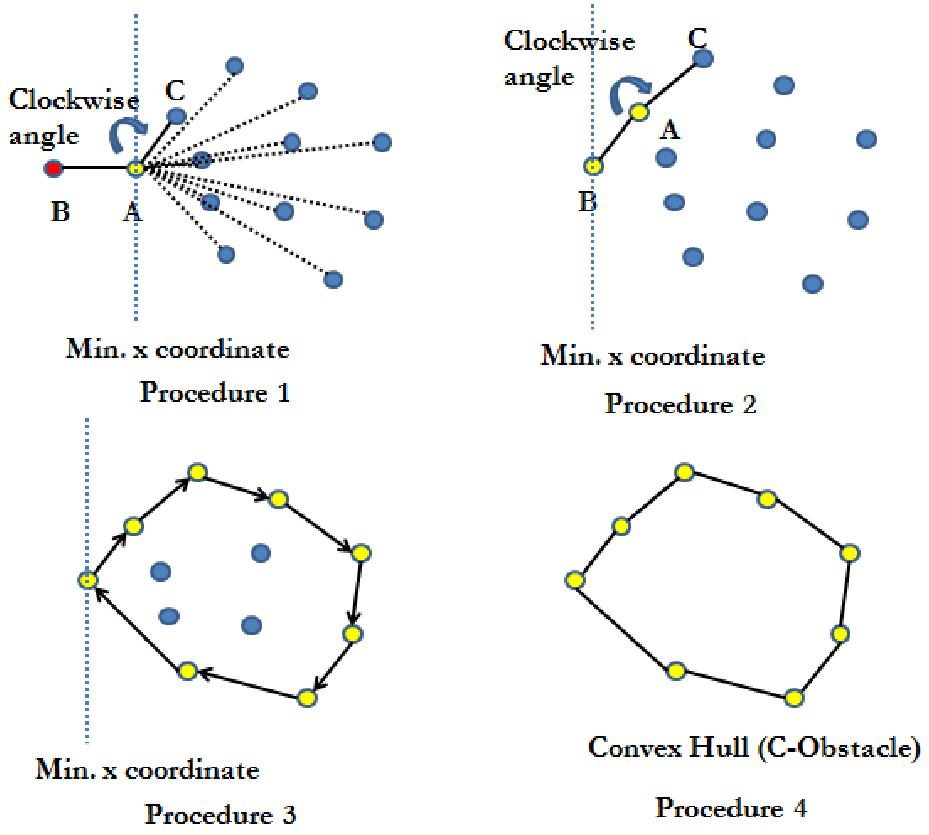


Figure 3.8 Creation of Convex Hull

In "Procedure 2", node *A* in "Procedure 1" will be renamed as node *B*, and the node which gives the smallest *BAC* in "Procedure 1" will be renamed as node A in "Procedure 2". Then the method described in "Procedure 1" is repeated and a new node *C* can be detected and marked. By continuing this method, all the vertices of the convex hull, shown as the yellow nodes in "Procedure 3", will be detected one by one until the starting node (node *A* in "Procedure 1") is reached. Eliminating other nodes inside and connecting the outer nodes gives the shape of the convex hull, which is the C-Obstacle. The algorithm of calculating any clockwise *BAC* angle (θ) through the process of finding convex hull is introduced in the following paragraph.

It is assumed that the coordinates of node *A* are (x_A, y_A), of node *B* are (x_B, y_B), and of node *C* are (x_C, y_C). Then two vectors are created, $\overrightarrow{AB}$ and $\overrightarrow{AC}$, by Equations (5) and (6). The dot product and cross product of $\overrightarrow{AB}\ and\ \overrightarrow{AC}$ are calculated using Equations (7) and (8), and the norms are determined by Equations (9) and (10). Equations (11) and (12) give the values of $cos\,\theta$ and $sin\,\theta$.

$$\overrightarrow{AB} = (x_B - x_A, y_B - y_A) \tag{5}$$

$$\overrightarrow{AC} = (x_C - x_A, y_C - y_A) \tag{6}$$

$$\overrightarrow{AB} \bullet \overrightarrow{AC} = (x_B - x_A) \times (y_C - y_A) + (y_B - y_A) \times (x_C - x_A) \tag{7}$$

$$\overrightarrow{AB} \times \overrightarrow{AC} = [(x_B - x_A) \times (y_C - y_A) - (y_B - y_A) \times (x_C - x_A)]\vec{k} \tag{8}$$
($\vec{\mathrm{k}}$ is a unit vector, with the direction determined by righthand rule)

$$|\overrightarrow{AB}| = \sqrt{(x_B - x_A)^2 + (\,y_B - y_A)^2} \tag{9}$$

$$|\overrightarrow{AC}| = \sqrt{(x_C - x_A)^2 + (\,y_C - y_A)^2} \tag{10}$$

$$cos\,\theta = \frac{\overrightarrow{AB} \bullet \overrightarrow{AC}}{|\overrightarrow{AB}| \times |\overrightarrow{AC}|} \tag{11}$$

$$sin\,\theta = \frac{\overrightarrow{AB} \times \overrightarrow{AC}}{|\overrightarrow{AB}| \times |\overrightarrow{AC}|} \tag{12}$$

$$arccos\,\theta = \alpha \tag{13}$$
Equation (13) calculates the clockwise angle of $\overrightarrow{AB}\ and\ \overrightarrow{AC}$, ranging from 0° to 180°. If the clockwise angle is bigger than 180°, the clockwise angle equals to (360°-α). sine *a* will be checked with sine Θ to determine clockwise *BAC* angle by the following check condition: if sine *a* does not equal to sine Θ, the clockwise angle equals to α, otherwise, the clockwise angle equals to (360°-α).

Step#4 Loop Rotation: The key to generating C-Obstacles for different rotations is to find the shape of the lifted object with different rotations, depicted by the coordinates of its nodes, and then to repeat the algorithm described in "Step#1", "Step#2", and "Step#3". The algorithm to find the shape of the rotated lifted object is implemented based on the rotation matrix. In 2D, the rotation matrix (counterclockwise rotation) has the form shown in Equation (14). If the lifted object is in the shape of a rectangle and depicted by its four nodes *(x_i, y_i) (i=1,2,3,4)*, and the first node *(x_1, y_1)* is chosen as the representing point. Then the relevant coordinates of the vectors from the four nodes towards the representing point are calculated satisfying Equations (15) and (16). Also, if the rotation step (see section 3.2.1 for rotation step) equals to $\boldsymbol{\alpha}$, then *n* possible rotations of the lifted object exist. *n* is calculated by Equation (17). So for each possible rotation (*j*×*a*), the new coordinates for the lifted object are (x_{rel-i}, y_{rel-i}) *(i=1,2,3,4)*, calculated from Equation (18). After calculating the new coordinates of the lifted object, "Step#1", "Step#2", and "Step#3" are repeated to generate the C-Obstacle. So with *j* changes from 1 to *n*, *n* times of generations for C-Obstacles are looped though. After all the C-Obstacles are calculated, the checks of connections between nodes will be discussed in the following section.

$$R(\Theta) = \begin{bmatrix} cos\Theta & -sin\Theta \\ sin\Theta & cos\Theta \end{bmatrix} \tag{14}$$

$$x_{rel-i} = x_i - x_1 \; (i = 1,2,3,4) \tag{15}$$

$$y_{rel-i} = y_i - y_1 (i = 1,2,3,4) \tag{16}$$

$$n = \frac{360}{\alpha} \tag{17}$$

$$\begin{bmatrix} xnew_i \\ ynew_i \end{bmatrix} = \begin{bmatrix} \cos(j \times \alpha) & -sin(j \times \alpha) \\ sin(j \times \alpha) & cos(j \times \alpha) \end{bmatrix} \times \begin{bmatrix} x_{rel-i} \\ y_{rel-i} \end{bmatrix} (i = 1,2,3,4)(j = 1,2,3,4, \dots, n) \tag{18}$$

3.3 Methodology of Connection Checking

3.3.1 Connecting Nodes of the Same Layer

The generated C-Obstacles are represented by nodes on various layers, and this section focuses on the method of finding the connections between the nodes on the same layer. The corresponding methodology is to generate a visibility graph (Asano, 1985). The visibility graph should meet the following conditions: 1) Only the start point, end point, and the nodes of C-Obstacles can be connected; 2)Two nodes are connected if and only if the connecting line segment is an edge of a C-Obstacle or it does not intersect any C-Obstacle.

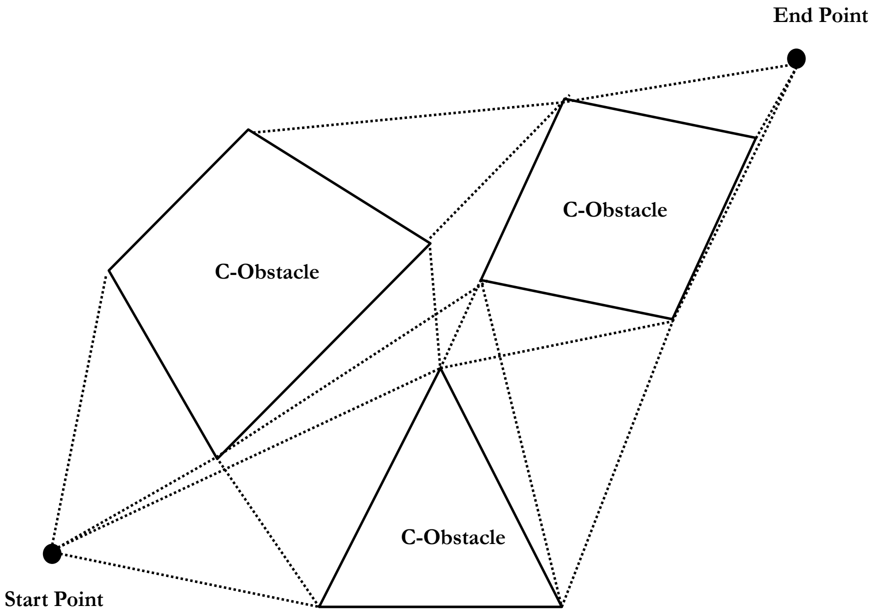


Figure 3.9 Visibility Graph

Figure 3.9 shows an example of a visibility graph, in which only the start point, end point, and the nodes of the C-Obstacles are considered for possible connections. The feasible paths, from the start point to the end point consist of the dash lines as well as the edges of the C-Obstacles. Among these paths, the one with the shortest distance is the shortest path. The basic problem when generating a visibility graph is to check whether two specific nodes (nodes of the generated C-Obstacle or start point or end point) can be connected or not. The situation that two nodes cannot be connected occurs when the connection collides with the C-Obstacle(s), which is checked by the following algorithm: Two nodes that are tested whether can be connected are first linked as a line. Then the line is checked with every edge of the C-Obstacles (the edge is also a line segment). If any intersection between the line and one of the edges exists, it means these two points cannot be connected, otherwise the connection is linked. So the core algorithm for building a visibility graph is to check the relationship of two line segments. The following paragraph will detail the algorithm mathematically.

It is assumed that two line segments are to be tested for intersection, L_1 and L_2. Each has two end points, p_1 and p_2 for L_1, and p_3 and p_4 for L_2. All four end points are denoted by $P_i = (x_i, y_i)$ $(i = 1, 2, 3, and\ 4)$. Vectors $(\vec{V_1}\text{ and }\vec{V_2})$ are created by Equations (19) and (20). $\vec{V_1}$and $\vec{V_2}$, represent the directions of the two line segments, L1 and L2. Equations (21) and (22) gives any two lines which parallel with $\vec{V_1}$ and $\vec{V_2}$ respectively. Equation (23) is used to check the intersection of *Line1* and *Line2*, and it can be rewritten as Equation (24). $\vec{V_1}$, $\vec{V_2}$, p_3 and $p_4$in Equation (24) are replaced by the values of their *x* and *y* coordinates, which gives Equations (25) and (26). These two equations can be written as Equation (27) by matrixes. At last, the values of α and β are calculated by Equation (28). If *a* and *β* are both between 0 and 1, it means that *L1* and *L2* intersect somewhere in the middle of both line segments; If any of them is bigger than 1 or less than 0, that means the two lines intersect somewhere on the extension of either *L1* or *L2*, or both; If two lines are parallel, there will not be any result for α and β. (Table 4.1)

$$\vec{V}_1 = (x_1 - x_2, y_1 - y_2) \tag{19}$$

$$\vec{V}_2 = (x_3 - x_4, y_3 - y_4) \tag{20}$$

$$Line\ 1 = p_2 + \alpha\vec{V}_1\ (\alpha\ is\ real\ number) \tag{21}$$

$$Line\ 2 = p_4 + \beta\vec{V}_2\ (\beta\ is\ real\ number) \tag{22}$$

$$p_2 + \alpha\vec{V}_1 = p_4 + \beta\vec{V}_2 \tag{23}$$

$$\alpha\vec{V}_1 - \beta\vec{V}_2 = p_4 - p_2 \tag{24}$$

$$\alpha(x_1 - x_2) - \beta(x_3 - x_4) = x_4 - x_2 \tag{25}$$

$$\alpha(y_1 - y_2) - \beta(y_3 - y_4) = y_4 - y_2 \tag{26}$$

$$\begin{bmatrix} x_1 - x_2 & x_4 - x_3 \\ y_1 - y_2 & y_4 - y_3 \end{bmatrix} \times \begin{bmatrix} \alpha \\ \beta \end{bmatrix} = [x_4 - x_2 \quad y_4 - y_2] \tag{27}$$

$$\begin{bmatrix} \alpha \\ \beta \end{bmatrix} = [x_4 - x_2 \quad y_4 - y_2] \times \frac{1}{|A|}A^* \tag{28}$$

$Where\ A = \begin{bmatrix} x_1 - x_2 & x_4 - x_3 \\ y_1 - y_2 & y_4 - y_3 \end{bmatrix};$

$A^* = \begin{bmatrix} y_4 - y_3 & -(x_4 - x_3) \\ -(y_1 - y_2) & x_1 - x_2 \end{bmatrix};$

$|A| = (x_1 - x_2) \times (y_4 - y_3) - (x_4 - x_3) \times (y_1 - y_2)$ ($|A| = 0$ if L_1 and L_2 are parallel)

Table 3.1 Intersection Checking Conditions

Relationships of two line segments	Conditions
Intersection	$(0 \le \alpha \le 1 \text{ and } 0 \le \beta \le 1)$
Parrallel	$(\lvert A\rvert = 0)$
Non-Intersection and Non-parallel	$\left(\alpha>1 \text{ or } 0>\alpha, \text{and } 0 \le \beta \le 1; \beta>1 \text{ or } 0>\beta, \text{and } 0 \le \alpha \le 1; \alpha>1 \text{ or } 0>\alpha, \text{and, } \beta>1 \text{ or } 0>\beta\right)$

3.3.2 Connecting Nodes between Layers

After finding connections among nodes on each layer, the next step is to connect the nodes between layers. The connection between two nodes on two different layers represents the rotation of the lifted object, and if these two nodes have different *x* and *y* coordinates, this connection also indicates a translation for the lifted object. However, the idea of layers converts the continuous rotating process of the lifted object to discrete rotations. One connection between any two layers represents a discrete motion of the lifted object. For each connection between two layers, the lifted object rotates from one angle to another, and meanwhile travels a distance (if there is horizontal movement), without considering the gradual rotating and horizontal traveling between the two layers. In other words, the initial and final orientations of the lifted object do not cause collision with obstacles. However, during the rotation, the collision may happen. Therefore, in order to minimize the risk of collision while performing the rotation step between two layers, it is assumed that only two nodes from successive layers can be connected, which guarantees the minimal rotation between layers. Also, the translation distance involved in any discrete motion between two layers is kept under a small unit. In addition, if two nodes from successive layers can be connected, a value is assigned to the connection as its length. Nonetheless, the value of the length has different unit since the motion between two successive layers involves a rotation.

Figure 3.10 represents the ideas from the previous paragraph. There are two successive layers, with nodes on layer *A* being marked as blue and nodes on layer *B* being marked as red. We try to find whether node 1 and node 2 can be connected. Layer *A* is overlaid on layer *B*, and the planar distance *k* between node 1 and node 2 can be calculated. If *k* is less than an accuracy parameter *a*, we consider this translation is safe and therefore node 1 and node 2 can be connected, otherwise, the connection is rejected. A length parameter is assigned as the value of the length for the connection of node 1 and node 2. In section 4.3, the user-defined parameters will be introduced relating to the length of the connection as well the accuracy parameter.

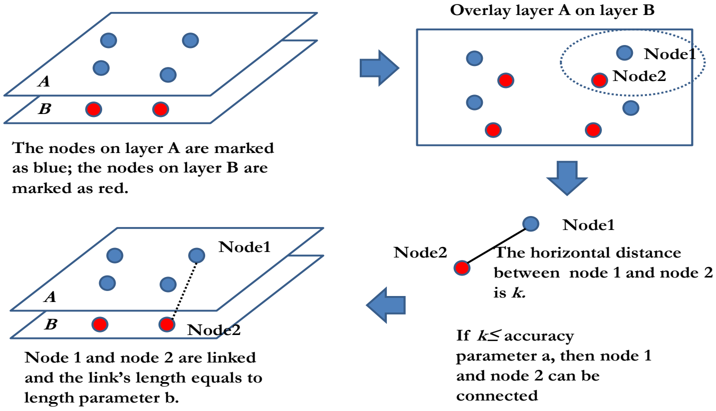

A
B
The nodes on layer A are marked as blue; the nodes on layer B are marked as red.
Overlay layer A on layer B
Node1
Node2
Node1
Node2
The horizontal distance between node 1 and node 2 is *k*.
If *k*≤ accuracy parameter a, then node 1 and node 2 can be connected
Node1
A
B
Node2
Node 1 and node 2 are linked and the link's length equals to length parameter b.

Figure 3.10 Process of Finding the Link of Two Nodes from Successive Layers

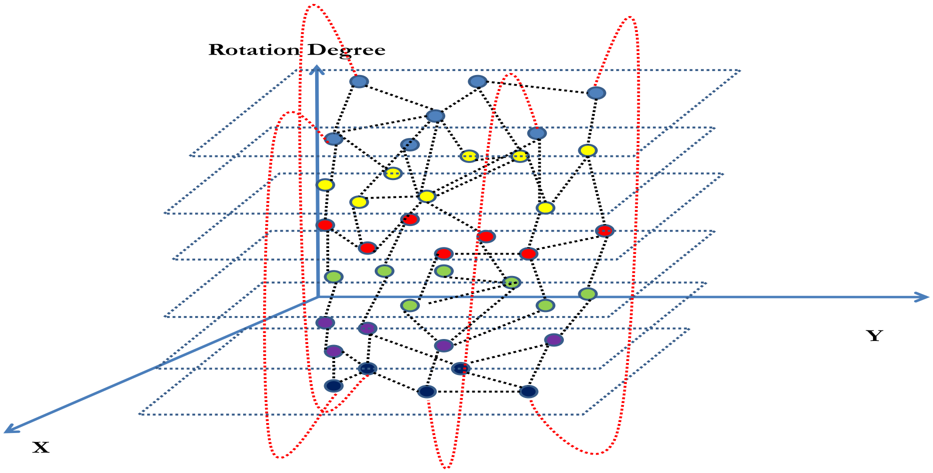


Figure 3.11 Connection Network

All the connections from same layer and between successive layers form a network as shown in Figure 3.11. In Figure 3.11, nodes with different colors represent nodes on different layers, and they are connected on the same layer or on successive layers (black dash lines). In addition, some red dash lines present the connections of the nodes between the bottom layer and the top layer. All the connections from each layer and between successive layers are stored in a m×m adjacency matrix *M* (Equation (28)), where C_{ij} is used to present the length of the connection between two nodes from the connection network (Figure 3.11). If two nodes (For example node 1 and node 2) can be connected, then C_{12} and C_{21} have the same value. The entries of the matrix that represent the connections that do not exist (for example, two nodes that cannot be connected) as well as the entries on the main diagonal are assigned an infinite real number.

$$\text{Adjacency}\ matrix\ M = \begin{bmatrix} C_{00} & C_{01} & C_{02} & & C_{0(m-2)} & C_{0(m-1)} & C_{0m} \\ C_{10} & C_{11} & C_{12} & \cdots & C_{1(m-2)} & C_{1(m-1)} & C_{1m} \\ C_{20} & C_{21} & C_{22} & & C_{2(m-2)} & C_{2(m-1)} & C_{2m} \\ & \vdots & & \ddots & & \vdots & \\ C_{(m-2)0} & C_{(m-2)1} & C_{(m-2)2} & & C_{(m-2)(m-2)} & C_{(m-2)(m-1)} & C_{(m-2)m} \\ C_{(m-1)0} & C_{(m-1)1} & C_{(m-1)2} & \cdots & C_{(m-1)(m-2)} & C_{(m-1)(m-1)} & C_{(m-1)m} \\ C_{m0} & C_{m1} & C_{m2} & & C_{m(m-2)} & C_{m(m-1)} & C_{mm} \end{bmatrix} \tag{28}$$

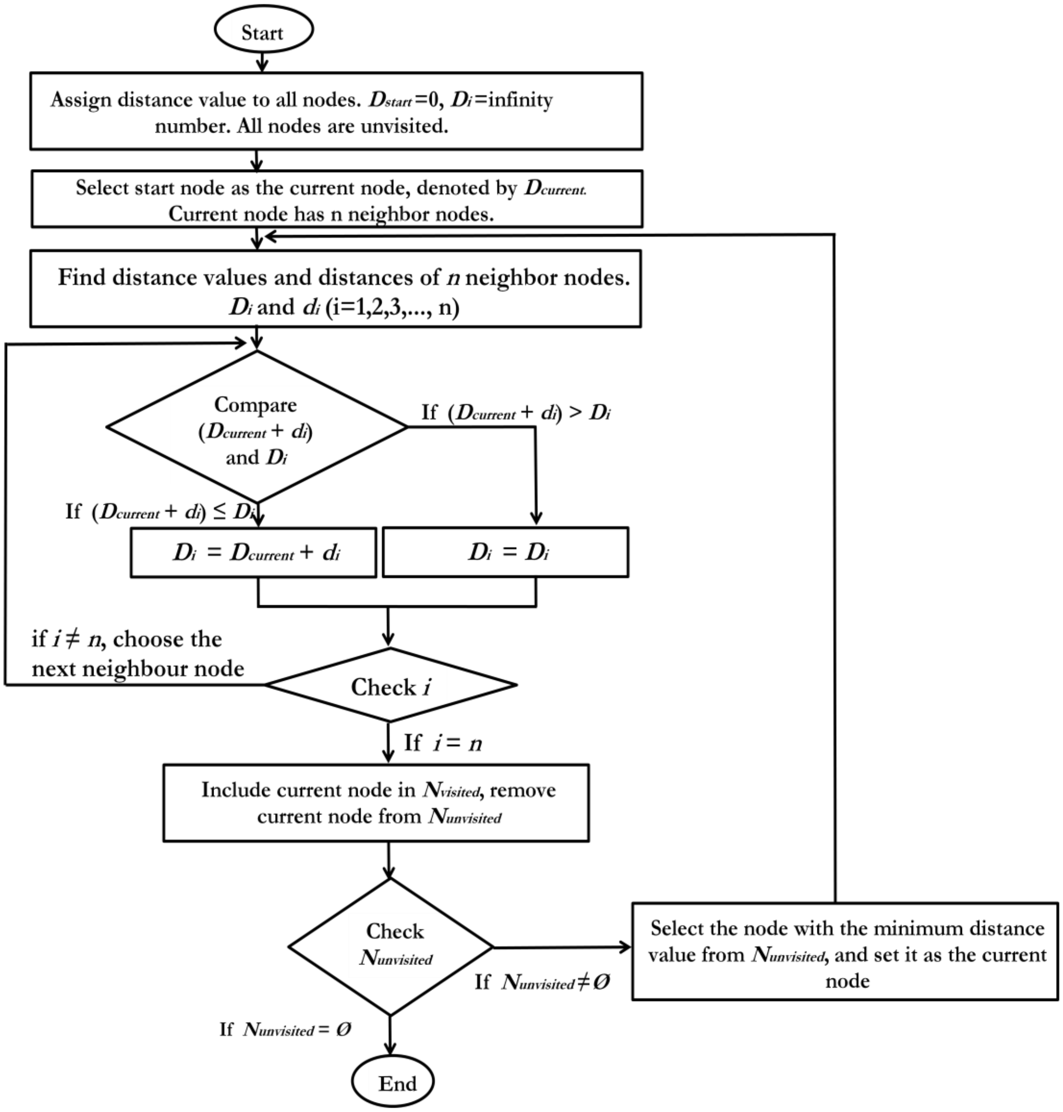
3.4 The Methodology of Finding a Feasible Path

Based on the adjacency matrix *M* (Equation (28)), the shortest path from the start node to the end node can be found by Dijkstra's algorithm. Dijkstra's algorithm has been proved efficient in many academic and practical fields. The basic idea of this method is to calculate the shortest distance from the start node to every other node (we name it as forward part), and then search the feasible path from the end point backwards (we call it backward part). The process of implementing the Dijkstra's algorithm forward part is described in Figure 3.12 and briefly reviewed here. After implementing the forward part, a distance array *N* (Equation (29)) is generated. In distance array N, L_{0j} *(j=0, 1, 2, 3, ..., n)* represents the distance from node 0 (start node) to node *j*.

$$Distance\ array\ N = \begin{bmatrix} L_{01} & L_{02} & L_{03} & L_{04} & \dots\dots & L_{0(n-3)} & L_{0(n-2)} & L_{0(n-1)} & L_{0n} \end{bmatrix} \quad (29)$$

After the Dijkstra's algorithm forward part, the shortest distance from the start node to every other node is found. By applying the Dijkstra's algorithm backward part, the shortest path can be obtained. The flowchart of the backward part is pictured in Figure 3.13. The results will be stored in the path array *P* in sequence. Equation (30) is an example of path array *P*. According to Equation (30), the shortest path is: the lifted object starts from node 0 (start node), travel through nodes (travel from node 4 to node 9 for example) following the sequence in which they are recorded, and finally reaches node 40 (end node).

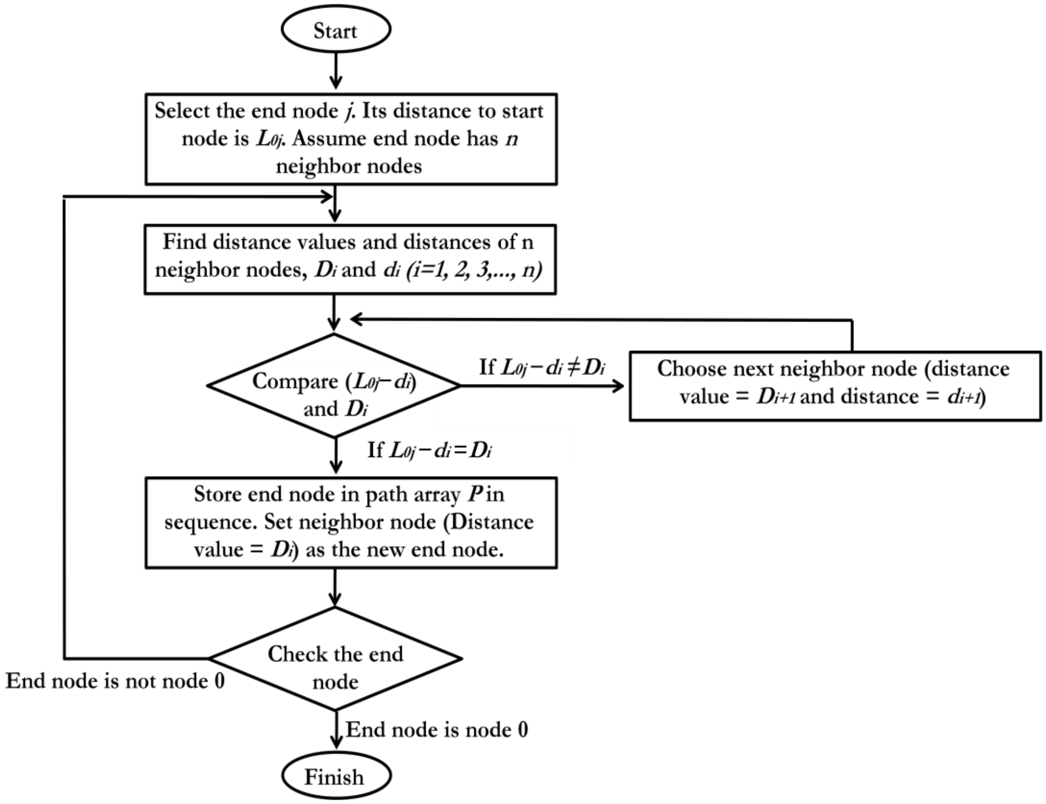
$$Path\ array\ P = \begin{bmatrix} 0 & 4 & 9 & 20 & \dots\dots & 19 & 25 & 37 & 40 \end{bmatrix} \quad (30)$$


Start
Assign distance value to all nodes. D_{start}=0, D_i=infinity number. All nodes are unvisited.
Select start node as the current node, denoted by $D_{current}$. Current node has n neighbor nodes.
Find distance values and distances of n neighbor nodes. D_i and d_i (i=1,2,3,..., n)
Compare ($D_{current}$ + d_i) and D_i
If ($D_{current}$ + d_i) > D_i
If ($D_{current}$ + d_i) ≤ D_i
D_i = $D_{current}$ + d_i
D_i = D_i
if $i \neq n$, choose the next neighbour node
Check i
If $i = n$
Include current node in $N_{visited}$, remove current node from $N_{unvisited}$
Check $N_{unvisited}$
If $N_{unvisited} \neq Ø$
Select the node with the minimum distance value from $N_{unvisited}$, and set it as the current node
If $N_{unvisited}$ = Ø
End

Note:

1. Current node: The node that is chosen to calculate the distance value of its neighbor nodes;
2. Distance value: The value of distance from the current node to node *i*, denoted by D_i ; Distance value of current node is denoted by $D_{current}$;
3. Neighbor nodes: The nodes whose distances to current node are not infinity in each loop;
4. Distance: Distance between current node and node *i*, denoted by d_i ;
5. Visited nodes: The set of calculated current nodes, denoted by $N_{visited}$;
6. Unvisited nodes: Assume that all the nodes are presented by set N_{all}, and the unvisited nodes are presented by set $N_{unvisited}$, which equals to ($N_{all} - N_{visited}$) ;
7. Start node: The position where the lifted object is picked in C-space;
8. End node: The position where the lifted object is located in C-space.

Figure 3.12 Flowchart of Forward Part of Dijkstra's Algorithm


Start
Select the end node *j*. Its distance to start node is L_{0j}. Assume end node has *n* neighbor nodes
Find distance values and distances of n neighbor nodes, D_i and d_i $(i=1, 2, 3,..., n)$
Compare $(L_{0j}-d_i)$ and D_i
If $L_{0j}-d_i \neq D_i$
Choose next neighbor node (distance value = D_{i+1} and distance = d_{i+1})
If $L_{0j}-d_i=D_i$
Store end node in path array *P* in sequence. Set neighbor node (Distance value = D_i) as the new end node.
Check the end node
End node is not node 0
End node is node 0
Finish

Note:

1. L_{0j}: The distance from node 0 (start node) to node *j* ;
2. Distance value: The value of distance from the current node to node *i*, denoted by D_i ;
3. Neighbor nodes: The nodes whose distances to end node are not infinity in each loop;
4. Distance: Distance between current node and node *i*, denoted by d_i ;
5. Path array P: shortest path (the results) stored in sequence from start node to end node.

Figure 3.13 Flowchart of Backward Part of Dijkstra's Algorithm

3.5 Summary and Conclusions

In this Chapter, the methodology and algorithm for this research has been presented. The method implemented to solve the crane heavy lift path planning is robotic motion planning. Generally, the significance of this method is to convert the shape of lifted object into a single point (a representing point), and find the traveling path for it. This method avoids checking the clearance for the entire shape of the lifted object. The implementation of this method can be divided into three parts. First, as discussed in section 3.2, the C-Obstacles are built based on the method obstacle growth. For one specific rotation of the lifted object, the corresponding C-Obstacles are generated and presented on one layer by the coordinates of their vertices, also known as nodes. Different layers are formed based on various rotations of the lifted object. Second, as described in section 3.3, the connections of the nodes from the same layer or from successive layers are detected. On each layer, two nodes can be connected if the connection does not collide with any C-Obstacle. For two nodes from successive layers, they can be connected only if the planar distance between them is small enough to meet the safety requirement. These found connections are stored in an adjacency matrix. Third, section 3.4 introduces the Dijkstra's Algorithm to search the shortest path based on the adjacency matrix. It consists of two parts, forward part as well as backward part. In forward part, the minimum distance between each node and the start node is calculated; the backward part searches the path from the end node to start node based on the distance values obtained from forward part.

Meanwhile, two assumptions are made for the connections between layers. First, when checking the connections of the nodes, only two nodes from successive layers can be connected. Second, two nodes from successive layers can be connected only if the planer distance between them meet the safety requirement. The rotations of the lifted object are considered as discrete steps between successive layers. So it raises a limitation that the rotating and moving process between two successive layers is unknown. However, these two assumptions simplify the continuous rotating processes, and reasonable considerations of these two assumptions are discussed in section 3.3.2. The implementation of the method and algorithms will be presented in the following chapter.

CHAPTER 4: IMPLEMENTATION PROCESS

This chapter presents the implementation process of the methodology stated in chapter 3. First, the structure of the developed system will be introduced. It is followed by three case studies to present the performance of the designed system.

4.1 Overview

As illustrated in Figure 4.1, the process starts with entering basic data and user-defined parameters. Following that, through communications between the database and the programming environment, the shortest path is calculated and entered into database table "tblSPNodes". Finally, the results are presented in a 2D graphical environment.

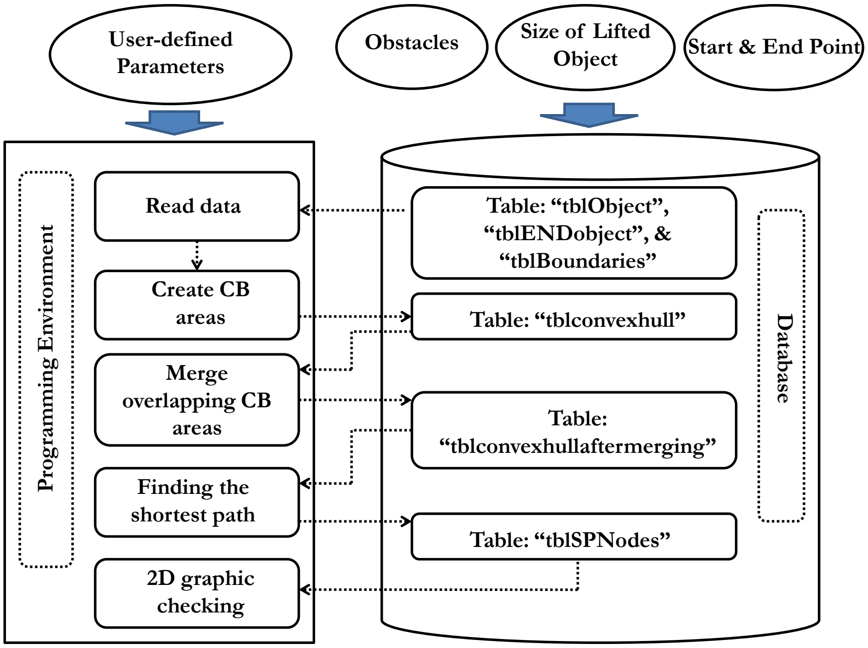
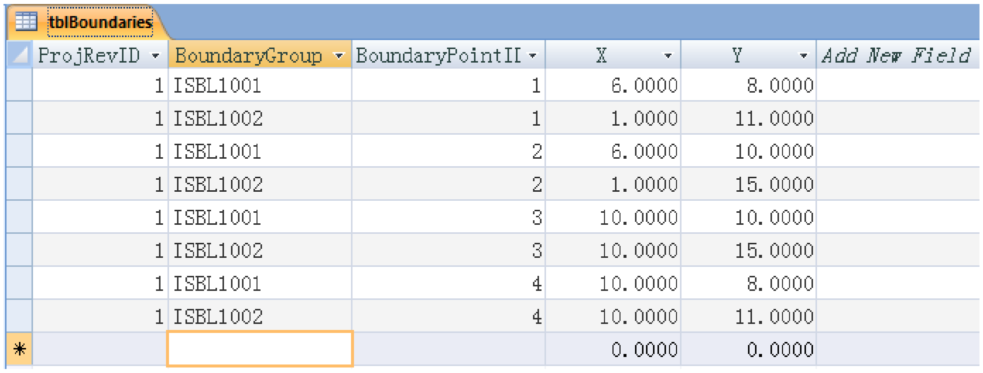


Figure 4.1 Implementation Process

4.2 Database Implementation

The database used in this research project has been constructed in Microsoft Access 2007. It consists of six tables, namely: (1) “tblBoundaries”; (2) “tblconvexhull”; (3) “tblconvexhullaftermerging”; (4) “tblENDobject”; (5) “tblObject”; and (6) “tblSPNodes”. These tables are further explained as the following:

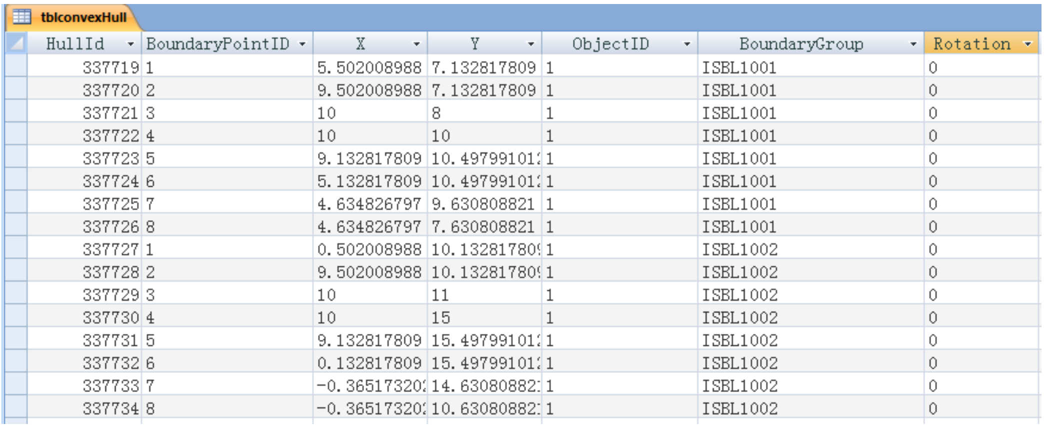
1. “tblBoundaries” (Figure 4.2) contains the data of the obstacles. “ProjRevID” is the project ID; “BoundaryGroup” defines the names of obstacles, for example “ISBL1001” is the first obstacle; “BoundaryPointID” presents the serial number of the nodes of obstacles; and “X” and “Y” define the coordinate of each node.



	ProjRevID	BoundaryGroup	BoundaryPointII	X	Y	Add New Field
	1	ISBL1001	1	6.0000	8.0000	
	1	ISBL1002	1	1.0000	11.0000	
	1	ISBL1001	2	6.0000	10.0000	
	1	ISBL1002	2	1.0000	15.0000	
	1	ISBL1001	3	10.0000	10.0000	
	1	ISBL1002	3	10.0000	15.0000	
	1	ISBL1001	4	10.0000	8.0000	
	1	ISBL1002	4	10.0000	11.0000	
*				0.0000	0.0000	

Figure 4.2 Structure of Database Table “tblBoundaries”

2. “tblconvexhull” (Figure 4.3) contains the data of C-Obstacles. “HullId” is the data ID; “ObjectID” defines which lifted object is chosen to create the C-Obstacles; “BoundaryGroup” presents the names of the C-Obstacles; “BoundaryPointID” presents the serial number of nodes for different C-Obstacles; “Rotation” determines the rotation angle of the lifted object that each layer represents; and “X” and “Y” define the coordinate of each node.

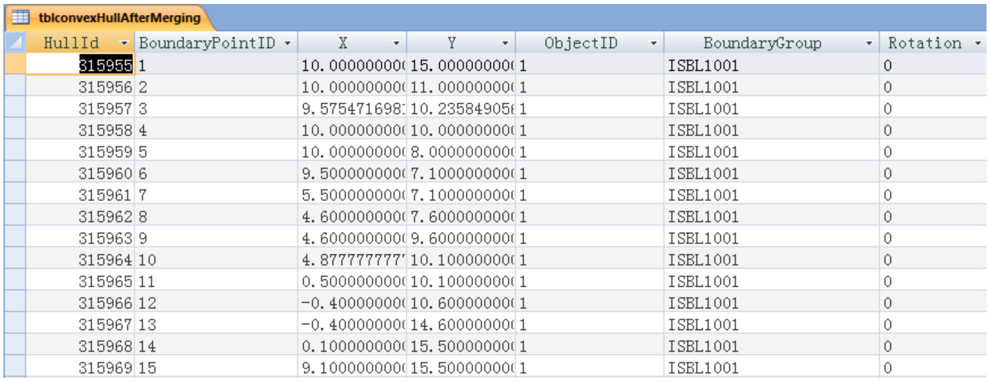


tblconvexHull

HullId	BoundaryPointID	X	Y	ObjectID	BoundaryGroup	Rotation
337719	1	5.502008988	7.132817809	1	ISBL1001	0
337720	2	9.502008988	7.132817809	1	ISBL1001	0
337721	3	10	8	1	ISBL1001	0
337722	4	10	10	1	ISBL1001	0
337723	5	9.132817809	10.49799101	1	ISBL1001	0
337724	6	5.132817809	10.49799101	1	ISBL1001	0
337725	7	4.634826797	9.630808821	1	ISBL1001	0
337726	8	4.634826797	7.630808821	1	ISBL1001	0
337727	1	0.502008988	10.13281780	1	ISBL1002	0
337728	2	9.502008988	10.13281780	1	ISBL1002	0
337729	3	10	11	1	ISBL1002	0
337730	4	10	15	1	ISBL1002	0
337731	5	9.132817809	15.49799101	1	ISBL1002	0
337732	6	0.132817809	15.49799101	1	ISBL1002	0
337733	7	-0.36517320	14.63080882	1	ISBL1002	0
337734	8	-0.36517320	10.63080882	1	ISBL1002	0

Figure 4.3 Structure of Database Table "tblconvexhull"

3. "tblconvexhullaftermerging" (Figure 4.4) contains the data of the merged C-Obstacles. As discussed before, when creating C-Obstacles, C-Obstacles may overlap. After merging those which overlap, the data of the new C-Obstacles are entered into "tblconvexhullaftermerging". The structure of this table is the same as table "tblconvexhull".



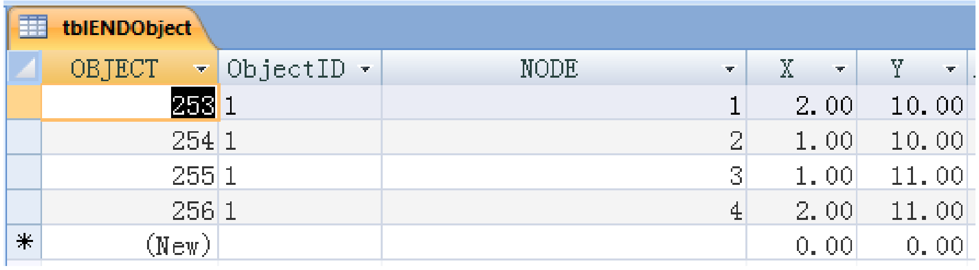
tblconvexHullAfterMerging

HullId	BoundaryPointID	X	Y	ObjectID	BoundaryGroup	Rotation
315955	1	10.00000000	15.00000000	1	ISBL1001	0
315956	2	10.00000000	11.00000000	1	ISBL1001	0
315957	3	9.575471698	10.23584905	1	ISBL1001	0
315958	4	10.00000000	10.00000000	1	ISBL1001	0
315959	5	10.00000000	8.000000000	1	ISBL1001	0
315960	6	9.500000000	7.100000000	1	ISBL1001	0
315961	7	5.500000000	7.100000000	1	ISBL1001	0
315962	8	4.600000000	7.600000000	1	ISBL1001	0
315963	9	4.600000000	9.600000000	1	ISBL1001	0
315964	10	4.877777777	10.10000000	1	ISBL1001	0
315965	11	0.500000000	10.10000000	1	ISBL1001	0
315966	12	-0.40000000	10.60000000	1	ISBL1001	0
315967	13	-0.40000000	14.60000000	1	ISBL1001	0
315968	14	0.100000000	15.50000000	1	ISBL1001	0
315969	15	9.100000000	15.50000000	1	ISBL1001	0

Figure 4.4 Structure of Database Table "tblconvexhullaftermerging"

4. "tblENDobject" and "tblObject" (Figure 4.5 and 4.6) contain the data of the start point and end point of the lifted object. "OBJECT" is an auto number generated by the database system, which is not used in the calculations;

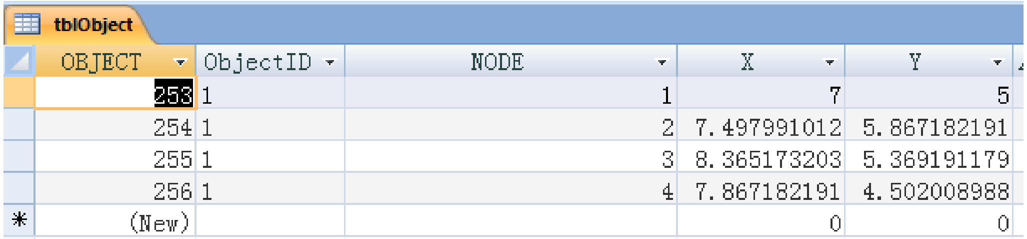
"ObjectID" determines the lifted object ID; "NODE" presents the vertices of the lifted object; "X" and "Y" state the coordinates of each node.



tblENDObject

OBJECT	ObjectID	NODE	X	Y
253	1	1	2.00	10.00
254	1	2	1.00	10.00
255	1	3	1.00	11.00
256	1	4	2.00	11.00
(New)			0.00	0.00

Figure 4.5 Structure of Database Table "tblENDObject"

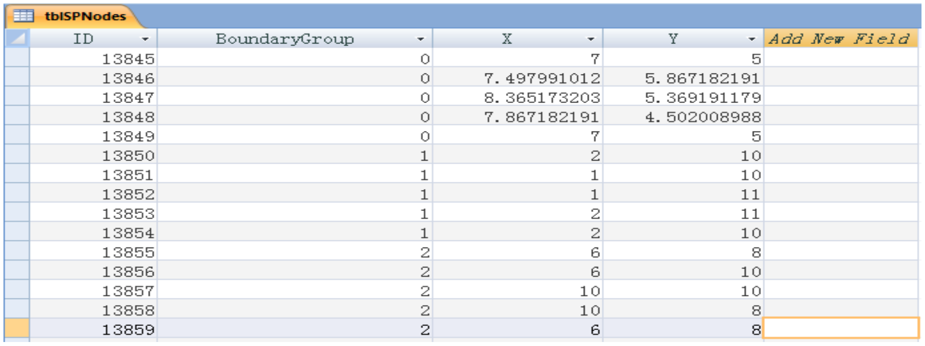


tblObject

OBJECT	ObjectID	NODE	X	Y
253	1	1	7	5
254	1	2	7.497991012	5.867182191
255	1	3	8.365173203	5.369191179
256	1	4	7.867182191	4.502008988
(New)			0	0

Figure 4.6 Structure of Database Table "tblObject"

5. "tblSPNodes" (Figure 4.7) contains the results (shortest path). "ID" is autonumber generated by the database system; "BoundaryGoup" indicates the results. "X" and "Y" describe the coordinates of the different nodes.



tblSPNodes

ID	BoundaryGroup	X	Y	Add New Field
13845	0	7	5	
13846	0	7.497991012	5.867182191	
13847	0	8.365173203	5.369191179	
13848	0	7.867182191	4.502008988	
13849	0	7	5	
13850	1	2	10	
13851	1	1	10	
13852	1	1	11	
13853	1	2	11	
13854	1	2	10	
13855	2	6	8	
13856	2	6	10	
13857	2	10	10	
13858	2	10	8	
13859	2	6	8	

Figure 4.7 Structure of Database Table "tblSPNodes"

4.3 The Implementation of the Program

The developed algorithm has been constructed in Microsoft Visual Basic.Net (VB.Net), which communicates with Microsoft Access. The program contains two interfaces, namely a control interface and a 2D graphical interface (Figure 4.8). In the control interface, the user can input parameters ("Rotation Step", "Rotation-Translation Limitation", "Weight of Connection between Layers", and "Relative Rotation"). These four parameters are defined as following:

- Rotation step: Defines the rotation interval of the lifted object. If the "Rotation step" is *x*, the number of layers equals to (360°/*x*).
- Rotation-Translation Limitation: Defines an accuracy parameter (as mentioned in section 3.3.2). If the planar distance between two nodes from successive layers is less than this parameter, then these two nodes can be connected;
- Weight of Connection between Layers: Determines the length of the corresponding connection if two nodes from successive layers can be connected.
- Relative Rotation: Determines the relative angular position of the lifted object at the end point with respect to the start point. This angle is in degrees and is measured in the counter-clockwise direction.

By clicking the "Convex Hull" button, the C-Obstacles are generated and the overlapping C-Obstacles are handled. Then the "Shortest Path" button carries out the function of searching for the shortest path by implementing Dijkstra's algorithm. After the optimal solution is found, the "Graphics" button activate interface 2, the graphical interface, which then presents the optimization results step by step through the clicking of "Steps" button.

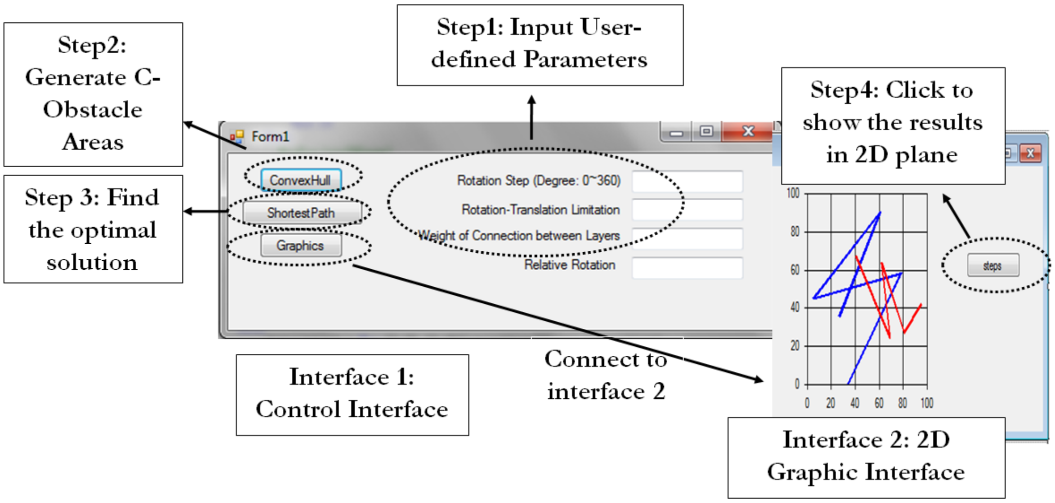

Step2: Generate C-Obstacle Areas
Step1: Input User-defined Parameters
Step4: Click to show the results in 2D plane
Form1
ConvexHull
ShortestPath
Graphics
Rotation Step (Degree: 0~360)
Rotation-Translation Limitation
Weight of Connection between Layers
Relative Rotation
Step 3: Find the optimal solution
100
80
60
40
20
0
0
20
40
60
80
100
steps
Interface 1: Control Interface
Connect to interface 2
Interface 2: 2D Graphic Interface

Figure 4.8 Control Interface and 2D Graphical Interface in VB.Net

4.4 Case Studies

In this section, three case studies are presented. Case one is a simple example to demonstrate the basic idea of how the designed system works. Case two and case three present more complex scenarios, in which complicated travelling paths are presented.

4.4.1 Case One

Case one contains two obstacles, ISBL1001 and ISBL1002, as shown in Figure 4.9 (Coordinates are plotted in Excel). The coordinates of ISBL1001, ISBL1002, the lifted object's start point and end point are given in Table 4.1. The coordinates in Table 4.1 are first input into the MS Access database "Case1.accdb", and the calculations are conducted in the programming system. During the calculations, multiple communications occur between database and programming environment (Figure 4.10).

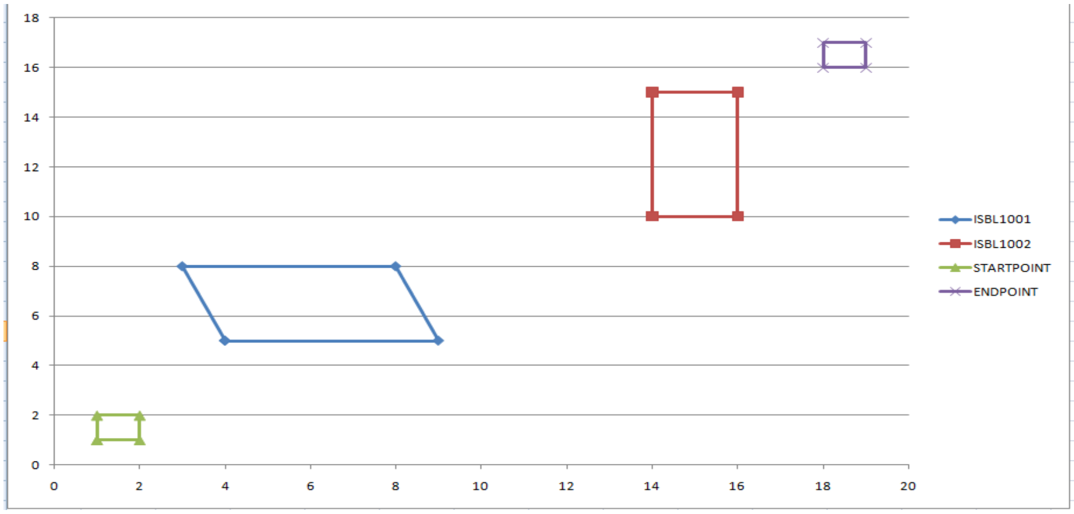
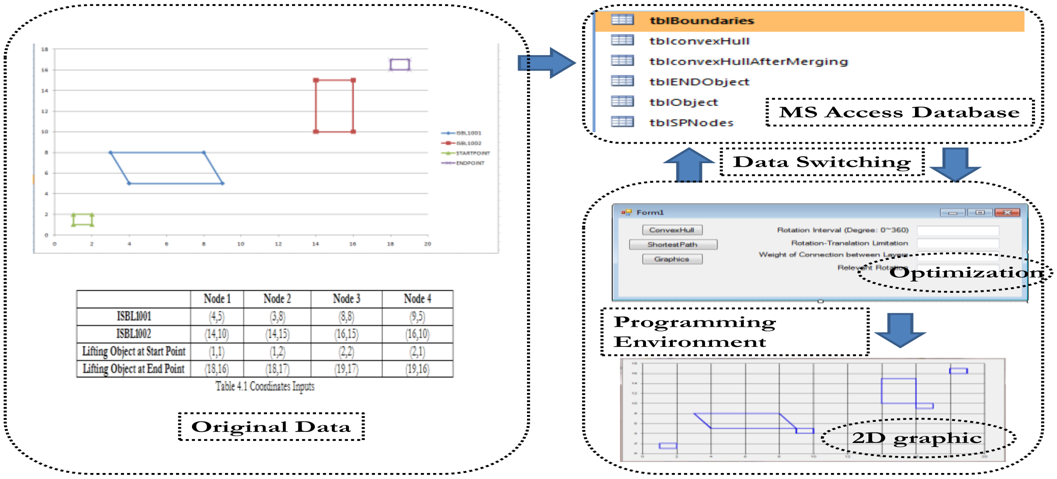


Figure 4.9 Case One Scenario

Table 4.1 Coordinates Case One Inputs

	Node 1	Node 2	Node 3	Node 4
ISBL1001	(4,5)	(3,8)	(8,8)	(9,5)
ISBL1002	(14,10)	(14,15)	(16,15)	(16,10)
Lifted Object at Start Point	(1,1)	(1,2)	(2,2)	(2,1)
Lifted Object at End Point	(18,16)	(18,17)	(19,17)	(19,16)

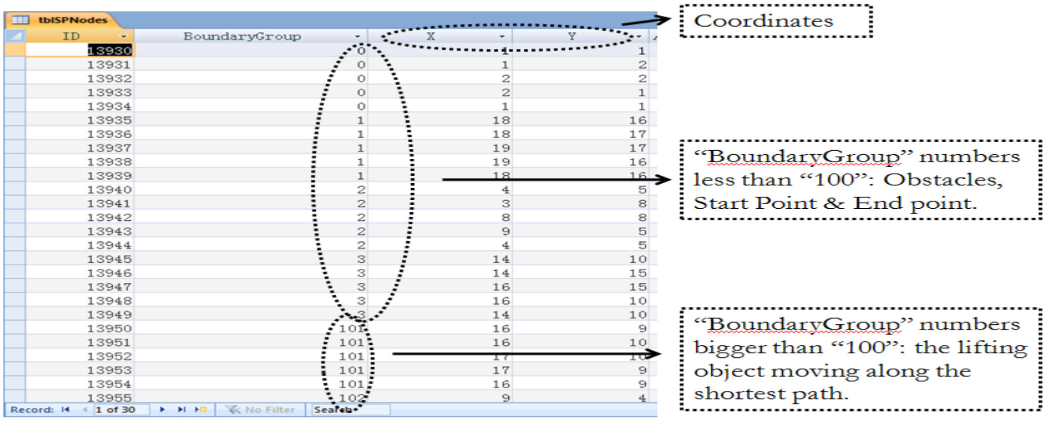


	Node 1	Node 2	Node 3	Node 4
ISBL1001	(4,5)	(3,8)	(8,8)	(9,5)
ISBL1002	(14,10)	(14,15)	(16,15)	(16,10)
Lifting Object at Start Point	(1,1)	(1,2)	(2,2)	(2,1)
Lifting Object at End Point	(18,16)	(18,17)	(19,17)	(19,16)

Table 4.1 Coordinates Inputs

Figure 4.10 Implementation of Case One

After the shortest path is found, the results are written in database table "tblSPNodes" (Figure 4.11). Those polygons whose "BoundaryGroup" numbers are less than 100 are obstacles, the start point, and end point, while those which are bigger than 100 represent the lifted object on the shortest path. These results as stated in Figure 4.11 can be plotted automatically in the 2D graphical interface (Figure 4.12). By clicking the button "Steps", the system shows the path step by step.



tblSPNodes

ID	BoundaryGroup	X	Y
13930	0	1	1
13931	0	1	2
13932	0	2	2
13933	0	2	1
13934	0	1	1
13935	1	18	16
13936	1	18	17
13937	1	19	17
13938	1	19	16
13939	1	18	16
13940	2	4	5
13941	2	3	8
13942	2	8	8
13943	2	9	5
13944	2	4	5
13945	3	14	10
13946	3	14	15
13947	3	16	15
13948	3	16	10
13949	3	14	10
13950	101	16	9
13951	101	16	10
13952	101	17	10
13953	101	17	9
13954	101	16	9
13955	102	9	4

Record: 1 of 30 No Filter Search

Coordinates

"BoundaryGroup" numbers less than "100": Obstacles, Start Point & End point.

"BoundaryGroup" numbers bigger than "100": the lifting object moving along the shortest path.

Figure 4.11 Results in Database

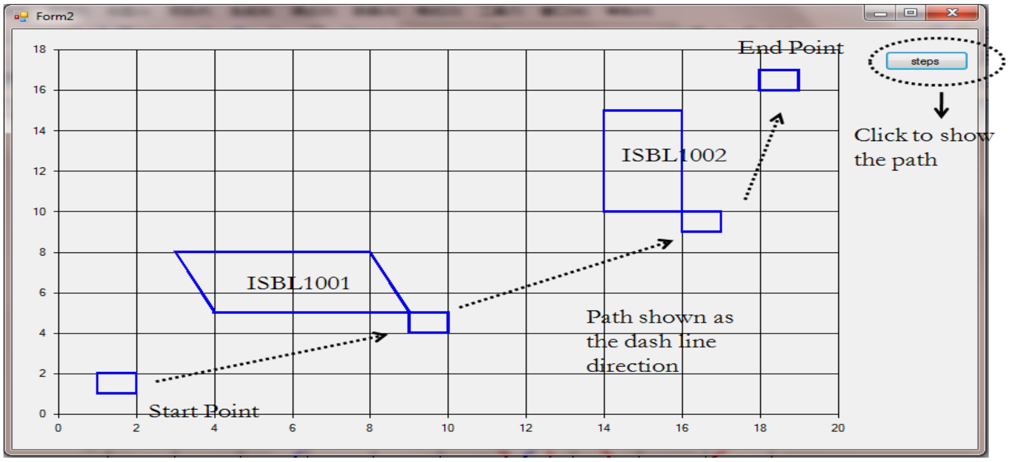


Figure 4.12 2D Graphic Presentation of Case One Results

Case one successfully presented a simple motion of a lifted object. However, the two obstacles are far from each other, and the traveling and rotation through the path are quite simple. The second case will demonstrate a more complicated scenario in which obstacles are very close, and the travelling path is more complex.

4.4.2 Case Two

Case two brings out a more congested workspace. Table 4.2 shows the coordinates of the obstacles and lifted object at pick point and end point. The coordinates in Table 4.2 can be plotted in Excel which results in Figure 4.14. It can be easily seen that the shortest path is the red dashed line from the start point to the end point.

Table 4.2 Coordinates Case Two Inputs

	Node 1	Node 2	Node 3	Node 4
ISBL1001	(1,1)	(1,10)	(5,10)	(5,1)
ISBL1002	(1,12)	(1,16)	(5,16)	(5,12)
ISBL1003	(6.2,7)	(6.2,9)	(11,9)	(11,7)
Lifted Object at Start Point	(7,5)	(7,6)	(9,6)	(9,5)
Lifted Object at End Point	(1,10)	(1,11)	(3,11)	(3,10)

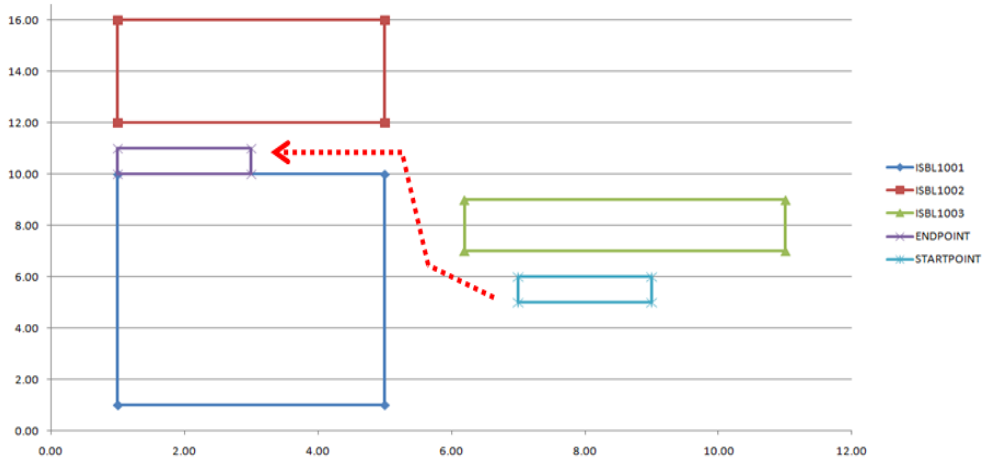
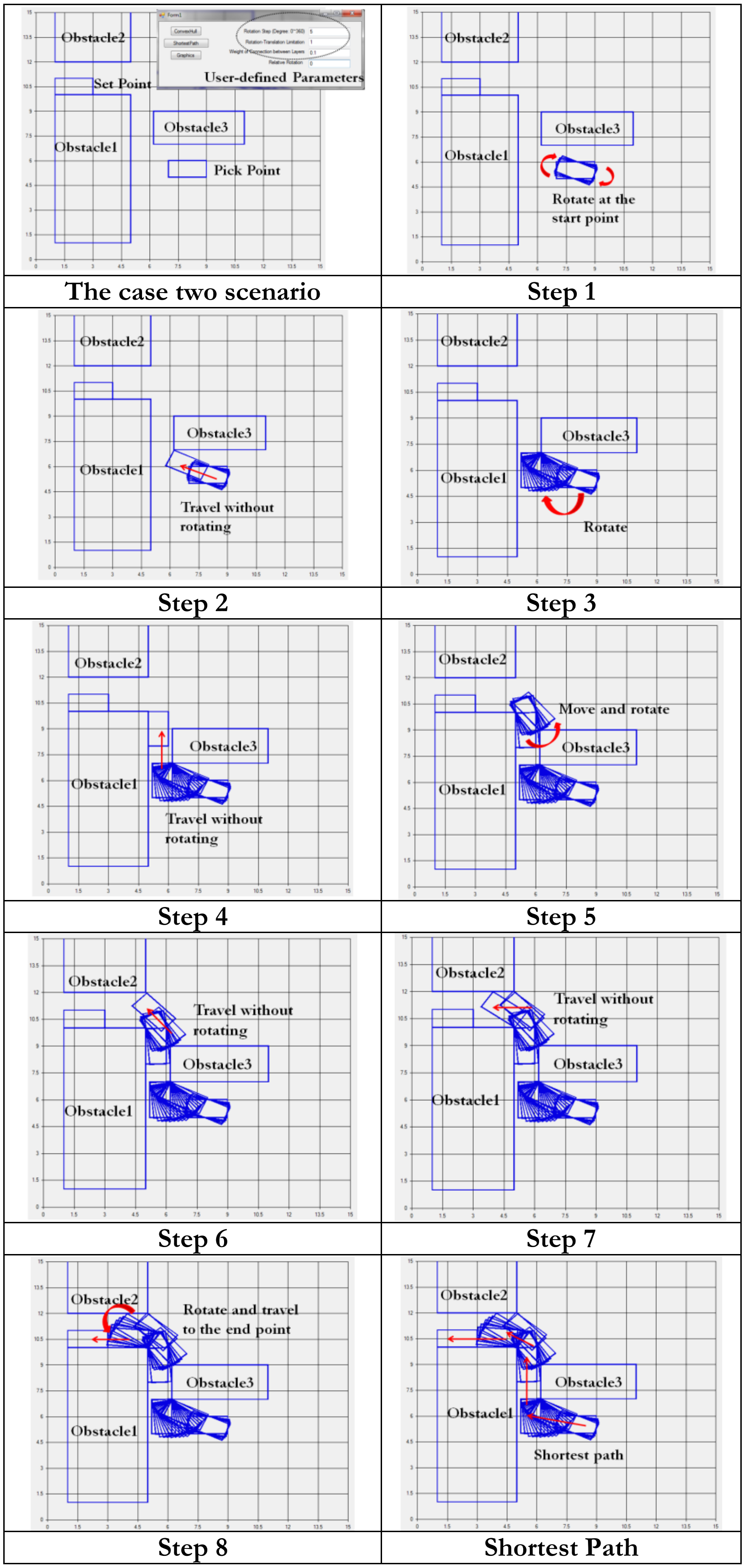


Figure 4.13 Case Two Scenario

The user-defined parameters are entered as inputs (See Figure 4.14 "Case Two Scenario"). After running the program, a detailed path is obtained (Figure 4.16). In order to show the path clearly, in Figure 4.16, eight steps are presented to show the moving process. Also, if the user does not want too many occurrences of rotation of the lifted object during the path, he can reduce the value of “Rotation-Translation Limitation” or increase the “Weight of Connection between Layers”. By reducing the value of “Rotation-Translation Limitation”, fewer nodes are connected between successive layers; by increasing the value of “Weight of Connection between Layers”, the length of the link of two nodes from successive layers extends and when the shortest path is searched, the algorithm will automatically eliminate the path which involves more rotations. Figure 4.15 shows the test when reducing the “Rotation-Translation Limitation” to 0.2 and Figure 4.16 shows the test when increasing the “Weight of Connection between Layers” to 5. The shortest path follows the direction of the red arrow line in both figures and it can be seen that by reducing “Rotation-Translation Limitation” and increasing “Weight of Connection between Layers”, the rotation of lifted object is avoided and the path is recreated.



The case two scenario

Step 1

Step 2

Step 3

Step 4

Step 5

Step 6

Step 7

Step 8

Shortest Path

Form1

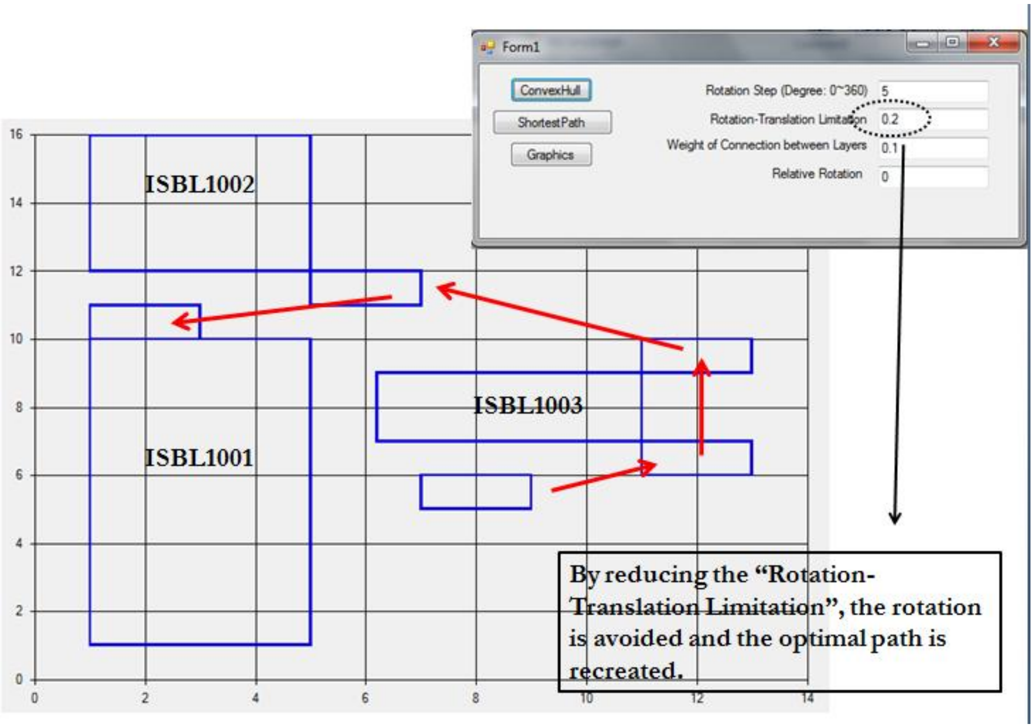
Rotation Step (Degree: 0~360) 5

Rotation-Translation Limitation 1

Weight of Connection between Layers 0.1

Relative Rotation 0

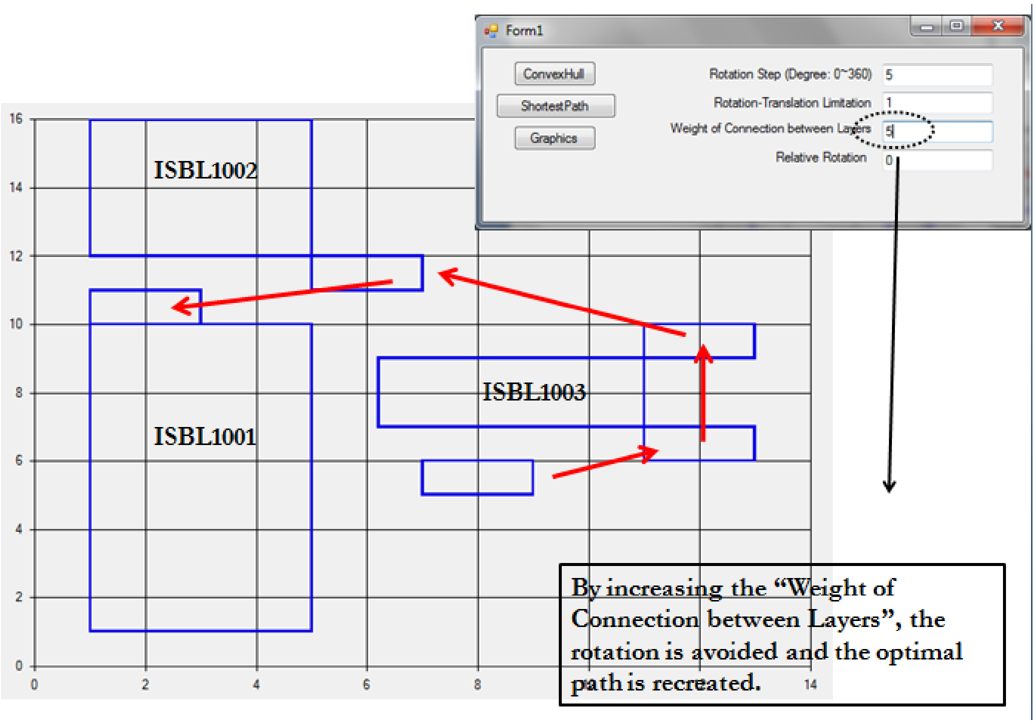
Figure 4.14 2D Graphic Presentation of Case Two Results



Rotation Step (Degree: 0~360) 5
Rotation-Translation Limitation 0.2
Weight of Connection between Layers 0.1
Relative Rotation 0

By reducing the "Rotation-Translation Limitation", the rotation is avoided and the optimal path is recreated.

Figure 4.15 Sensitivity Test One



Rotation Step (Degree: 0~360) 5
Rotation-Translation Limitation 1
Weight of Connection between Layers 5
Relative Rotation 0

By increasing the "Weight of Connection between Layers", the rotation is avoided and the optimal path is recreated.

Figure 4.16 Sensitivity Test Two

4.4.3 Case Three

In case three, the only possible path involves a much more sophisticated maneuver by the crane which elaborates on the capacity of the developed algorithm. The coordinates of the obstacles and lifted object at its pick point and place point are given in Table 4.3 and plotted in Figure 4.17. Then the coordinates are entered into the database, and after running the program, the results are plotted in 2D graphic interface (Figure 4.18). The difficulty for this moving is that the path between ISBL1001 and ISBL1003 is narrow, and the lifted object must be rotated and then moved through it (Step 1 to Step 5 in Figure 4.18). After the lifted object travels through the narrow path, it is rotated again and moved to its end point (Step 6 to Step 10 in Figure 4.18). The entire path is also shown in Figure 4.18.

Table 4.3 Coordinates Case Three Inputs

	Node 1	**Node 2**	**Node 3**	**Node 4**
ISBL1001	(-4, 7)	(-4, 10)	(5, 10)	(5, 7)
ISBL1002	(1, 11.5)	(1, 13)	(5, 13)	(5, 11.5)
ISBL1003	(6.2, 7)	(6.2, 10)	(11.5, 10)	(11.5, 7)
ISBL1004	(-4, 3)	(-4, 5.5)	(5, 5.5)	(5, 3)
Lifted Object at Start Point	(2, 5.5)	(2, 6.5)	(4, 6.5)	(4, 5.5)
Lifted Object at End Point	(1, 10)	(1, 11)	(3,11)	(3,10)

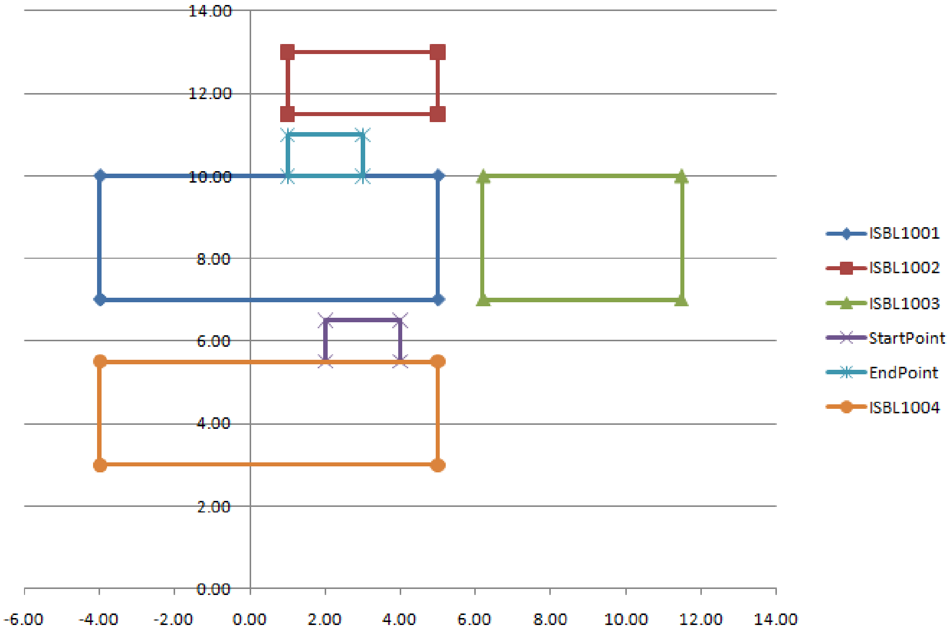
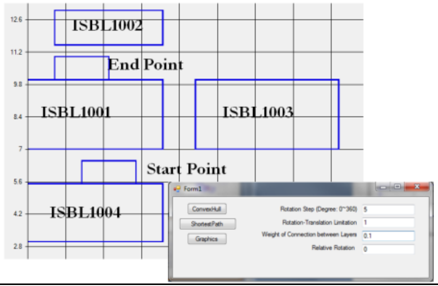
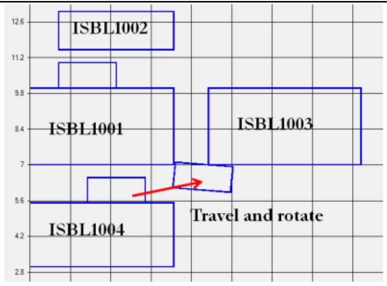


Figure 4.17 Case Three Scenario

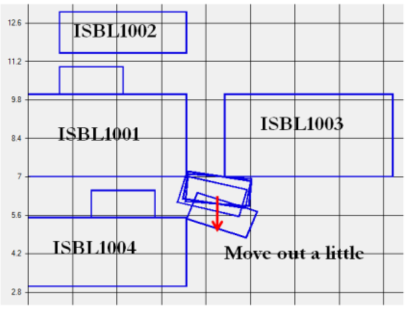


Rotation Step (Degree: 0~360) 5
Rotation-Translation Limitation 1
Weight of Connection between Layers 0.1
Relative Rotation 0

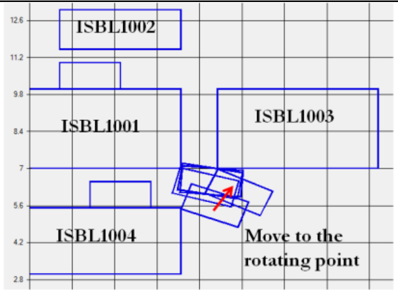
Case Three Scenario



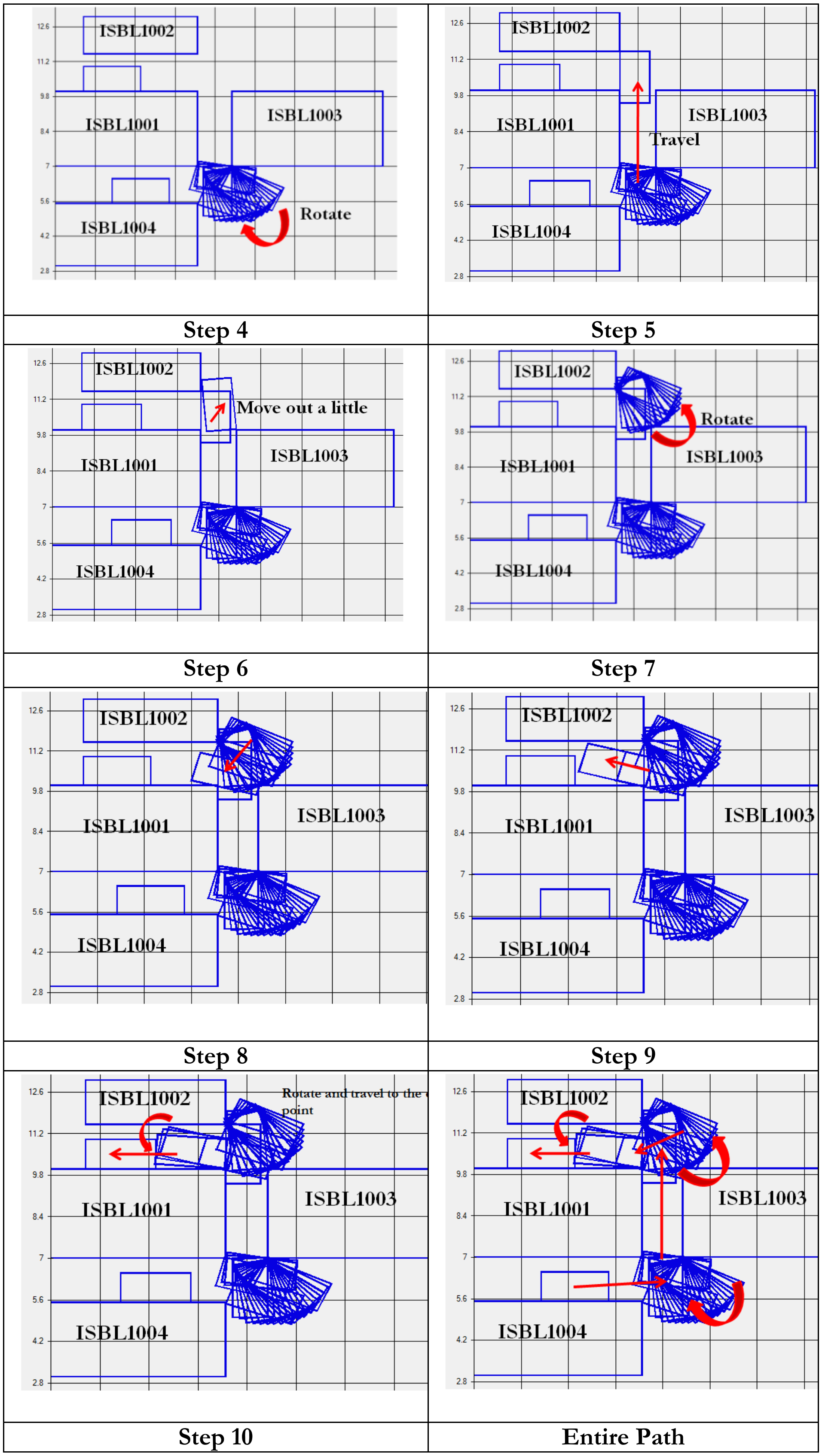
Step 1



Step 2



Step 3



Step 4

Step 5

Step 6

Step 7

Step 8

Step 9

Step 10

Entire Path

Figure 4.18 2D Graphic Presentation of Case Three Results

4.5 Conclusions and Discussions

This system successfully implements a robotic motion planning methodology to solve crane heavy lift path planning problem. In particular, it considers the lifted object as a planar mobile robot and builds the C-Space for it. It can automatically generate the shortest path for the lifted object and present the results in a 2D graphical environment. Path feasibility and sensitivity can be tested by controlling user-defined parameters.

In addition, according to the visibility graph as discussed in section 3.3.1, only the start point, end point, and the vertices of C-Obstacles can be connected. So the lifted object either travels from one node to another in the C-Space, or from one node to another node on the edge of the same C-Obstacle. Between the two nodes, the connection is a straight line. Therefore, this method guarantees that from the start point to the end point, the found path is the shortest. Meanwhile, three cases presented in this chapter present the path without collisions and the generated paths are reasonable based on the layout situation and inputs. By varying the user-defined parameters, the feasibility of different paths is shown. To validate the results, we adopt the method that the lengths of different paths are calculated manually, and then the path with the shortest distance is compared with the results generated by the system. It has been proved that the generated paths match the results of manual calculations.

CHAPTER 5: CONCLUSIONS & RECOMMENDATIONS

This chapter summarizes the contributions of the research work, along with the limitations of this proposed method, and gives the recommendations for future work.

5.1 General Conclusions

With the utilization of cranes, the efficiency of construction has been improved dramatically in recent decades, with increased productivity and quality, and savings of costs and time. Today's construction sites are generally congested, so heavy lift planning before and during the construction process is very important for the success of crane heavy lifts. Heavy lift planning contains several subtasks, one of which is path planning. However, manual path planning is prone to errors. This research is motivated by the need to develop a decision support system for the user in order to automate heavy lift path planning. This thesis has described the development of an automated system for heavy lift path planning, and by implementing robotic motion planning methodology. The developed program is implemented using Microsoft Visual Basic.Net and Microsoft Access database.

The user starts by inputting the coordinates of the obstacles and lifted object into the database, and through communications with the database, the shortest path is generated in the program and presented in a 2D graphical environment. The system has proven to be effective in circumventing potential collisions of the heavy lift path planning.

5.2 Research Contributions

The developed system in this research can potentially benefit the current crane heavy lift planning in many aspects. The contributions are summarized below:

- A new approach for motion planning of heavy lift cranes: The existing methods (Ali et al 2005; Sivakumar et al. 2003; Reddy et al. 2002) have chosen the crane as the manipulator, considered its degrees of freedom, and generated the configuration space for the crane. No research has been conducted with the lifted object viewed as the manipulator (mobile robot) in solving the construction crane path planning problem. The developed system generates the C-Obstacle for the obstacle based on the shape of the lifted object, automating the crane lift path planning process;
- Automated generation of the shortest and obstacle free path for crane: The developed system can automate the process of heavy lift path planning. According to the layout information in the database, the system automatically finds the shortest path for the lifted object from its start configuration to end configuration.
- Introduction of new user-defined parameters to control the aspects of the generated path: With user-defined parameters, the user can define the rotation step of the lifted object, and control the frequency of the occurrences of the rotation for the lifted object through its moving trajectory;
- 2D graphical path presentation interface: The 2D graphical path presentation interface can automatically plot the found shortest path and show its details.

5.3 Research Limitations

The limitations of the proposed solution are discussed as follows. First, the developed system considers only the 2D planar layout of the construction site. Second, the found path is composed of separate nodes, and between two nodes is a straight line. Therefore, the entire path may not be a curve and is not applicable for crane swing operation. In this case, the user has to modify the path to a curve manually. Third, the developed solution is designed to take care of only convex obstacles. For concave obstacles, the user has to break them down to small convex obstacles and then use the program.

5.4 Recommendations for Future Work

This developed system can be a foundation for many path-finding problems, such as the followings:

- The current 2D planar path planning system can be extended to a 3D path planning system. The elevations of the workspace may be taken into consideration;
- Crawler crane can lift the lifted object and travel to its destination. The developed methodology can be extended to develop a system to study the feasibility of moving the crane on site while carrying the lifted object from pick point to set point;
- The 2D crane path planning system can be expanded into a 3D environment by considering the elevation of the obstacle and lifted object;
- A 3D animation model can be built by using 3D Studio Max, for the purpose of assessing and validating the proposed algorithm.

- Analysis to determine the optimal buffers for lifted object and obstacles: various factors should be considered in order to guarantee the safety of the moving path for the lifted object. For example, wind (weather factor) may cause the swing of the lifted object during its movements. One of solutions for the wind factor may be adding a buffer around the lifted object or obstacles so that the generated paths tend to be safer. Further research may aims to develop algorithms to determine the size of the buffers considering these safety factors.
- 4D implementations: Construction project is related to schedule, and the layout changes as the project proceeds, which may affect the generated path planning. Therefore, in order to consider the schedule while ensure the correctness of the generated path planning, 4D technology is proposed as a future research recommendation, which integrates the 3D graphical model with the construction schedule.

Bibliography

- Al-Hussein, M. (1999). "An integrated system for crane selection and utilization." Ph.D. thesis, Department of Building Civil & Environmental Engineering, Concordia University, Montreal, Canada.
- Al-Hussein, M., Alkass, S., and Moselhi, O. (2000). "D-CRANE: Database system for utilization of cranes." *Canadian Journal of Civil Engineering*, 27, 1130-1138.
- Al-Hussein, M., Alkass, S., and Moselhi, O. (2005). "Optimization algorithm for selection and on-site location of mobile cranes." *Journal of Construction Engineering and Management*, ASCE, 131(5), 579-590.
- Ali, M. S., Babu, N. R., and Varghese, K., (2005). "Collision free path planning of cooperative crane manipulators using genetic algorithm." *Journal of Computing in Civil Engineering*, 19 (2), 182–193.
- Asano, T., Guibas, L., Hershberger, J., and Imai, H. (1985). "Visibility-polygon search and Euclidean shortest path." *The 26th Symposium on Foundations of Computer Science*, Portland, Oreg., October 21-23, 155-164.
- Aurenhammer, F. (1991). "Voronoi diagrams—A survey of fundamental geometric data structure." *ACM Computer Survey*, 23, 3 (Sept.), 345-405.
- Bajaj, C., and Kim, M.S. (1990). "Generation of configuration space obstacles: Moving algebraic surfaces."*International Journal of Robotics Research*, 9(1), 92-112.
- Boles, W.W., Maxwell, D.A., Scott, W.D., Heermann, P.D., Yarborough, T., and Underwood, J. (1995). "Construction automation and robotics. Pathway to implementation." *Journal of Construction Engineering and Management*, 121(1), 143-152.

- Brost, R.C. (1989). "Computing metric and topological properties of configuration-space obstacles", *International Conference Robotics Automation 1989*, 1, 170-176.

- Cheng, M.-Y., Liang, Y., Wey, C.-M., and Chen, J.-C. (2001). "Technological enhancement and creation of a computer-aided construction system for the shotcreting robot." *Automation in Construction*, 10, 517–526.

- Collier, E. and Fischer, M., (1995) "Four-dimensional modeling in design and construction", *CIFE Technical Report*, No. 101, Stanford University, Stanford, CA.

- Everett, J.G., and Slocum, A.H. (1994). "Automation and robotics opportunities: Construction versus manufacturing." *Journal of Construction Engineering and Management*, 120(2), 443-452.

- Faltings, B. (1987). "Qualitative kinematics in mechanisms", *Proceedings of IJCAI-87*, Milan, Italy, 436-442.

- Gouzènes, L. (1984). "Strategies for solving collision-free trajectories problems for mobile and manipulator robots", *International Journal of Robotics Research*, 3(4), 51.

- Hanna, A. S. and Lotfallah, W. B. (1999). "A fuzzy logic approach to the selection of cranes." *Automation in Construction*, 8(5), 597-608.

- Hasan, S., Al-Hussein, M., Hermann, U. H., and Safouhi, H. (2010) "Interactive and dynamic integrated module for mobile cranes supporting system design." *Journal of Construction Engineering and Management*, ASCE, 136(2), 179-186.

- Hwang, Y.K., and Ahuja, N. (1992). "Gross motion planning- A survey", *ACM Computing Surveys*, 24(3), 219-291

- Kamat, V. and Martines, J. (2001). "Visualization simulated construction operations in 3D." *Journal of Computing in Civil Engineering*, 15(4), 329-337.
- Kang, S. C., and Miranda, E., (2006). "Planning and visualization for auto-mated robotic crane erection processes in construction". *Automation in Construction*, 15 (4), 398–414.
- Kangari, R., and Halpin, D.W. (1989). "Potential robotics utilization in construction." *Journal of Construction Engineering and Management*, ASCE, 115(1), 126-143.
- Kim, S.-K., Russel, J. S., and Koo, K.-J. (2003). "Construction robot path-planning for earthwork operations." *Journal of Computing in Civil Engineering*, 17(2), 97–104.
- Latombe, J.C. (1991). "Robot motion planning." Kluwer Academic Publishers, Norwell, MA.
- Lozano-Pérez, T. (1983). "Spatial planning: A configuration space approach." *IEEE Transactions on Computers*, 32(2), 108-119.
- Ma, Z., Shen Q., and Zhang, J. (2005). "Application of 4D for dynamic site layout and management of construction projects." *Automation in Construction*, 14 (2005), 369-381.
- Mahalingam, A., Kashyap, R., and Mahajan, C. (2009). "An evaluation of the applicability of 4D CAD on construction projects." *Automation in Construction*, 19(2010), 148-159.
- Manrique, J.D., Al-Hussein, M., Telyas, A., and Funston, G. (2007). "Constructing a complex precast tilt-up-panel structure utilizing an optimization

model, 3D CAD and animation." *Journal of Construction Engineering and Management*, ASCE, 133(3), 199-207.

- Reddy, H. R., Varghese, K., (2002). "Automated path planning for mobile crane lifts." *Computer-Aided Civil and Infrastructure Engineering*, 17 (6), 439–448.
- Sandor, G.N., Erdman, A.G. (1984). "Advanced mechanism design", Prentice-Hall Inc.. Vol. I and II.
- Skibniewski, M.J., and Russell, J.S. (1989). "Robotic applications to construction." *Cost Engineering*, 31(6), 10-18.
- Tam, C.M., Tong, K.L., and Chan, K.W. (2001) "Genetic algorithm for optimizing supply location around tower crane." *Journal of Construction Engineering and Management*, 127 (4), 315– 321.
- Tam, C.M. and Tong, T.K.L. (2003). "GA-ANN Model for Optimizing the Location of Tower Crane and Supply Points for High-Rise Public Housing Construction." *Construction Management and Economics*, 21, 257-266.
- Sawhney, A. and Mund, A. (2002). "Adaptive Probabilistic Neural Network-based Crane Type Selection System." *Journal of Construction Engineering and Management*, ASCE, 128(3), 265-273.
- Shapira, A., Lucko, G., and Schexnayder, C. J. (2007). "Cranes for building construction projects." *Journal of Construction Engineering and Management*, ASCE., 1339, 690–700.
- Shapiro, H., Shapiro, J., and Shapiro, L. (1999). "Cranes and Derricks." 3rd Edition, McGraw-Hill, ISBN 0-07-057889-3.

- Sivakumar, PL., Varghese, K., and Babu, N.R. (2003). "Automated path planning of cooperative crane lifts using heuristic search." *Journal of Computing in Civil Engineering*, ASCE, 17(3), 197-207.
- Varghese, K., Dharwadkar, P., Wolfhope, J., and O'Connor, J.T. (1997). "A heavy lift planning system for crane lifts." *Microcomputers in Civil Engineering*, 12 (1997), 31-42.
- Warszawski, A. (1990a). "Expert system for crane selection construction." *Journal of Construction Management and Economics*, 8, 179-190.
- Warszawski, A. (1990b). "Industrialization and robotics in building." Harper & Row, New York, N.Y.
- Warszawski, A., and Navon, R. (1998). "Implementation of robotics in buildings: Current status and future prospects." *Journal of Construction Engineering and Management*, ASCE, 124(1), 31-41.
- Zhang, P., Harris, F. C., Olomolaiye, P. O., and Holt, G. D. (1999). "Location optimization for a group of tower cranes." *Journal of Construction Engineering and Management*, ASCE, 125(2), 115-112.