Massively parallelized GA based optimal path planning for single and dual crane lifting in complex industrial environments

Cai, Panpan

2016

Cai, P. (2016). Massively parallelized GA based optimal path planning for single and dual crane lifting in complex industrial environments. Doctoral thesis, Nanyang Technological University, Singapore.

https://hdl.handle.net/10356/68893

https://doi.org/10.32657/10356/68893

NANYANG
TECHNOLOGICAL
UNIVERSITY

MASSIVELY PARALLELIZED GA BASED OPTIMAL PATH PLANNING FOR SINGLE AND DUAL CRANE LIFTING IN COMPLEX INDUSTRIAL ENVIRONMENTS

CAI PANPAN

SCHOOL OF MECHANICAL AND AREOSPACE ENGINEERING

2016

Massively Parallelized GA Based Optimal Path Planning for Single and Dual Crane Lifting in Complex Industrial Environments

Cai Panpan

School of Mechanical and Aerospace Engineering

A thesis submitted to the Nanyang Technological University
in partial fulfilment of the requirement for the degree of
Doctor of Philosophy

2016

Abstract

Heavy lifting is an important task in petrochemical and pharmaceutical plants. It is frequently conducted during the time of plant construction, maintenance shutdown, and new equipment installation. Mobile cranes are lifting machines widely used in a variety of industries. The two primary issues that industries concern are safety and productivity. Accidents may happen in work sites of mobile cranes due to various reasons such as lack of operation knowledge, lack of safety awareness, lack of information about the environment, inadequate guidance, and wrong calculations in lifting. These factors may also influence the productivity by wasting time, energy and resources in unnecessary operations or stoppages. Computer-aided Lift Planning (CALP) for mobile cranes is an effective and efficient tool highly desired by industries.

This research aims to develop a new CALP system for automatic lift planning in complex industrial environments such as petrochemical and pharmaceutical plants, and construction sites. The research focuses on the lifting path planning problems for single and cooperative dual mobile cranes in these complex environments. The lifting path planning takes inputs such as plant environments, mechanical and positioning information of cranes, and start & end lifting configurations to generate optimal lifting paths by evaluating costs and risks involved. In this research, the single-crane and dual-crane lifting path planning are both formulated as multi-objective nonlinear optimization problems with multiple implicit constraints. The objective is to optimize the energy costs, time costs and safety factors of the lifting paths under constraints such as collision avoidance, coordination, and operational limitations. To solve the optimization problems, two master-slave parallel genetic algorithm based path planners are designed and developed on Graphic Processing Units (GPUs) using CUDA programming. The genetic algorithms in the planners are customized for the lifting

path planning problems with their efficiency and search abilities improved. In order to handle complex environments, an image-based collision detection algorithm is developed to support the planners. The image-space parallel collision detection algorithm constructs multi-level depth maps for industrial environments and takes advantage of GPU parallel computing. Based on this algorithm, a hybrid C-space collision detection strategy is introduced to trade off the pre-processing and planning time for the planners. To reduce the computation time for continuous collision detection for the lifting target in dual-crane lifting path planning, triangle swept spheres are introduced to model the swept volumes. Finally, a lift planner cum crane simulator system is developed based on the collision detection algorithm and the path planners enhanced by a lexicographical goal programming strategy. This system can serve the purposes of automatic lift planning, interactive lift planning, and training, and thus improve the safety and productivity of lifting operations.

Acknowledgments

The author would like to express her sincere thanks and appreciation to her supervisor, Dr. and Associate professor Cai Yiyu (MAE), and co-supervisor, Dr. and Associate professor Zheng Jianmin (SCE) for their invaluable guidance, support and suggestions. Their knowledge, suggestions, and discussions help me to become a capable researcher. Their encouragement also helps me to overcome many difficulties encountered in my research.

The author also wants to thank the colleagues in the CAE Visualization Room, for their generous help. I want to thank Dr. Indhumathi Chandrasekaran for providing the crane simulation environment and technical suggestions. I want to thank for Mr. Huang Lihui's help on converting PDMS data and the help of Dr. Ma Yuewen, Dr. Wu Xiaoqun, Dr. Zhang Yuzhe and Mr. Chen Yong on scanning and processing data in point cloud form. I also want to thank the FYP student Zhang Tianci for developing the physical model crane. My gratitude also goes to Dr. Li Qing and Dr. Chen Wenyu for their friendship and support.

I am very grateful to the members of our research team in NTU. It is a pleasure to collaborate with my project team mates, particularly Dr. Indhumathi Chandrasekaran and Mr. Huang Lihui.

Last but not least, I want to thank my husband Zhang Juzheng for his constant companion and my family in China for their love and encouragement.

Contents

List of Figures

List of Tables

List of Abbreviations

CALP	Computer-aided Lift Planning
GPU	Graphic Processing Unit
CPU	Central Processing Unit
GPGPU	General Purpose Graphic Processing Unit
CUDA	Compute Unified Device Architecture
CD	Collision Detection
DCD	Discrete Collision Detection
CCD	Continuous Collision Detection
BV	Bounding Volume
AABB	Axis-Aligned Bounding Box
OBB	Oriented Bounding Box
BVH	Bounding Volume Hierarchy
BSP	Binary Spatial Partitioning
SV	Swept Volume
SSV	Sphere Swept Volume
PSS	Point Swept Volume
LSS	Line Swept Volume
LDI	Layered Depth Image
VOI	Volume of Interest
MDM	Multi-level Depth Map
RSS	Rectangle Swept Volume
TSP	Traveling Salesman Problem
PRM	Probabilistic Roadmap Method
RRT	Rapidly exploring Random Tree
GA	Genetic Algorithm
SGA	Simple Genetic Algorithm
MSPGA	Master-Slave Parallel Genetic Algorithm
LGP	Lexicographical Goal Programming
IK	Inverse Kinematics
FK	Forward Kinematics
C-Space	Configuration Space
ASV	Analytical Swept Volume
TSS	Triangle Swept Sphere
NASA	National Aeronautics and Space Administration

List of Notations

α_{LF}	Luffing angle
α_{SW}	Swinging angle
l_{HS}	Hoisting length
α_{LR}	Load rotation angle
s	Chromosome or string
L_s	Length of a string
L_p	Size of population
c	A node configuration in a string
e	An edge segment in a string
C	C-space of a crane
$F(s)$	Fitness value of a string s
$sc(s)$	Operation switching cost in string s
$d(s)$	Motion cost in string s
r_m	Mutation rate
r_c	Crossover rate
A	Swept frontier
R	Working radius
λ	Constant scaling factor
mo	Coordination violation number of nodes
me	Edge coordination violation number of edges
no	Collision violation number of nodes
nf	Edge collision violation number of cranes
nr	Edge collision violation number of the single-crane load
nl_i	Edge collision violation number of the dual-crane load
nc	Collision violation number of internal clearance
sc	Operation switching count
$<<< a, b, c >>>$	Launch parameters for CUDA kernel functions
$\{\ldots\}$	Sets
$(\ldots)$	Ordered sets
$\|\|\mathbf{x} - \mathbf{y}\|\|$	Distance between $\mathbf{x}$ and $\mathbf{y}$
$\|x\|$	Absolute value of x
$((v))_{xy}$	The 2D vector formed by the x and y coordinates of $\mathbf{v}$
$((v))_z$	The z coordinate of $\mathbf{v}$

Chapter 1

Introduction

1.1 Background

Heavy cranes are widely used by the petrochemical, pharmaceutical, construction, and other industries for critical lifts. To lift large and heavy loads, the capacity of cranes can reach up to thousand tons. Lifting operations work with potential accidents. The OSHA database [4] reported a total 3135 crane-related accidents in the US till the year 2013. Many of them caused human death. Lifting safety is thus utmost important. Many reasons may lead to crane safety problems at the work site including inadequate guidance, insufficient training, lack of safety awareness or poor site assessment. Any accident may cause substantial losses of money and time, and waste of other resources especially when fatalities are involved. Cost reduction is another major concern in the lifting industry. According to a rental rate survey conducted by Cranes & Access in 2011 [5], the average daily rental of a terrain crane with a capacity of 350 tons can cost approximately 8,394 US dollars. Electric power and fuel consumption also takes up a significant portion of the cost in lifting operations. For instance, the power consumption of the terrain crane LTM 1200 from Liebherr is 370 KW per hour [6]. Therefore, optimizing the time and energy cost in heavy lifting is highly desired. On the other hand, manpower cost is increasingly becoming a critical factor in lifting. A lifting team is responsible for lift planning which involves a complicated and sophisticated decision-making process. The team consists typically of one lifting supervisor or manager, one engineer, one crane operator, one or more signalmen, and one or more

riggers. Therefore, efficient lift planning is crucial for the safety and productivity of any related industries.

Heavy lift planning typically involves crane selection, crane setup planning and lifting path planning. For projects with multiple lifts, the planning also requires scheduling. If multiple cranes are used simultaneously, the interference between these cranes also need to be considered. If the plant or site contains dynamic objects, the process may also include replanning mechanisms to alter the lifting path according to the changed environment. There are different types of commonly used cranes: tower cranes, terrain cranes, crawler cranes and so on. Among them, tower cranes have the least Degree of Freedoms (DOFs) (4 or less) and crawler cranes have the most (up to 7 DOFs). In this thesis, the name "mobile crane" is used mostly for terrain cranes and sometimes crawler cranes. Lift planning is usually done manually, which can be error-prone and very time-consuming. Even with an experienced lifting team, it easily takes a few weeks to complete the entire planning procedure. For the reasons mentioned above, it is desired to have an effective and efficient solution for intelligent lift planning.

Computer-aided Lift Planning (CALP) uses computer simulations and intelligent algorithms to assist the lift planning process and thus improve the lifting safety and productivity. CALP systems can serve two purposes: lift planning for real-life projects and vocational training for crane operators. For training purpose, CALP systems simulate the manipulations and monitor the safety factors for trainees to practice lifting in virtual environments. For planning purpose, CALP systems either focus on interactive lift planning or automatic lift planning. Interactive lift planning enables users to define lifting paths through trials and errors. Automatic lift planning aims to compute feasible or optimal lifting plans automatically with minimum user interventions.

In prior studies, researchers have developed simulation-based CALP systems with various safety monitoring functions embedded. Many of them have also automated procedures in crane setup planning. Among these systems, some focus on interactive lift planning using 2D drawings. However, these drawings have to be combined with many other materials to feature specific lifting tasks, making these planners tedious to use. Other systems require users to construct site models manually using CAD

software. Such approaches result in simple and less accurate models and are thus more suitable for training purposes. Moreover, automatic lifting path planning has seldom been addressed by previous CALP systems.

Instead of using 2D drafting and conventional CAD models, modern industrial plants and buildings have their digital versions in intelligent management systems such as Plant Design Management Systems (PDMS), Smartplant, and Building Information Modeling (BIM) systems. These systems manage the comprehensive geometric, scheduling and cost information for plants and sites. The information can be utilized to produce complete and accurate models of the environment objects and lifting targets. Schedules and lifting task specifications might also be extracted from these commercial systems. For existing plants or sites that do not have digital versions, laser scanning is an alternative. Laser scanning technology has experienced rapid developments in recent years. By combining with geometric processing techniques, laser scanning can quickly produce accurate point clouds for large and complex environments. However, as the technologies mentioned above always generate huge data sets, previous CALP systems have not made use of them.

Lifting path planning is one of the core technologies supporting automatic lift planning which has rarely been implemented. Using the information such as crane specifications, crane setups, environments, and task requirements, lifting path planning outputs safe and optimized paths to guide lifting operations automatically or semi-automatically. Most lifting tasks are conducted by using a single crane with enough capacity required. In this case, lifting path planning considers the interactions of three parties: the crane, the lifting target, and the environment. The lifting target and the crane components (e.g. booms, jibs and hooks) usually follow nonlinear movements. During lifting, the lifted target should always be kept at a safe clearance of distance from the crane and the obstacles. Under these constraints, the operational costs of lifting should be minimized. Aside from using a single large crane to fulfill the task, dual-crane lifting is also a common strategy in heavy lifting when large cranes with sufficient capacities to lift the load are neither available nor allowed due to space and budget restrictions. Therefore, two cranes with relatively lower individual capacities are alternatively used as a cheaper solution to share the load of the lifting target. In

dual-crane lifting operations, the major crane and the assistant crane collaborate to maintain the equilibrium of the lifting target. The slings should be restricted as close as possible to the vertical direction during the lifting operations. When performing cooperative lifting, the safety risk increases dramatically due to the intensive synchronizations between the cranes, as well as the highly nonlinear movements of the lifting target. Therefore, it is much harder to determine optimal dual-crane lifting paths under these safety constraints.

Lifting path planning for both single crane and dual cranes have been investigated as stand-alone motion planning problems. Some previous studies have tried to produce feasible but not necessarily optimal lifting paths using fast search algorithms (see Section 2.4 for details). Other approaches apply global optimization algorithms like Genetic Algorithms (GAs) to obtain optimized lifting paths. Most studies still rely on conventional CAD models and are thus restrained in relatively simple cases. Real industrial plants in 3D are seldom used for lifting path planning in the literature. Global optimization based approaches also suffer from the computationally forbidden feature of GAs. Many practical issues in lifting path planning such as easiness of conduction and coordination of cooperative cranes in terms of sling angles have not been well considered.

1.2 Problem Statement

This research aims to develop a CALP system for automatic lift planning in complex industrial environments such as petrochemical plants, pharmaceutical plants and construction sites. Except developing simulation and interaction components, it mainly focuses on solving the path planning problems of single-crane and dual-crane lifting. The lifting path planning problems take inputs such as plant environments, crane mechanical data, crane setup locations, start and end configurations and produce outputs as optimal lifting paths via cost and risk evaluations. The aim is to optimize practical factors such as the energy cost, time cost, and human operation conformity of the lifting path under constraints of collision avoidance, dual-crane coordination, and crane operational limitations. Achieving these goals requires the system to handle efficiently the complex environments in digital formats such as PDMS and laser scanned

point clouds. Therefore, this research also introduces a unified representation of plant data and develops an efficient collision detection algorithm to handle the interactions between cranes and these environments.

When solving the multi-objective multi-constraint problems, path planning algorithms can be easily trapped into local optima or converge to sub-optimal solutions. To achieve highly optimized lifting paths, GA is chosen to build the targeted path planners taking advantage of its global optimization capability. Formulation, computational efficiency, and convergence are the three major challenges in solving the targeted lifting path planning problems.

Firstly, the lifting path planning problems are required to be formulated in a way that is compatible with GA's representations. Many prior algorithms use keyframe positions in the Euclidean spaces as chromosomes. This formulation of paths is suitable for point-shaped or rigid robots. However, for articulated robots, this approach makes it difficult to evaluate the manipulation costs. On the other hand, objectives and constraints in the lifting path planning problems need to be formulated taking into consideration of the performances, costs and safety factors of lifting paths. The safety constraints are determined by the unique kinematic and physical properties of the problems which are substantially different in the single-crane and dual-crane scenarios. The crane is represented as a manipulation robot in single-crane lifting. The movement of the lifting target (the end effector) is completely determined by the manipulation variables through forward kinematics. Formulating the single-crane problem requires defining problem-specific metrics for lifting paths to evaluate their costs and safeties quantitatively. In dual-crane lifting, the two cranes and the lifting target form a closed kinematic chain which is not able to be solved with forward kinematics [3]. Moreover, differed from other cooperative robots, the dual-crane system contains a cable-suspended target which is lifted by the slings of two cranes. Thus, the position and orientation of the lifting target are not only affected by the manipulation variables, but also by the forces, torques and mass distribution of the lifting target. Therefore, the formulation of dual-crane lifting path planning is much more difficult than the single-crane problem.

Efficiency is the another challenge in GA-based lifting path planning. The path planners need to conduct collision detection for all paths in each iteration. They are

also required to handle properly the continuous motions along the local paths between genes using sound algorithms to be described. This results in huge computational effort considering the nonlinear movements of the cranes and the lifting target. Moreover, the computational loads can be drastically increased if highly complex environments are involved. Thus, it is critical to find efficient representations of the environments as well as the nonlinear motions of the cranes and the lifting target. Parallelization may also improve the computational efficiency of GA.

One more challenge is the search ability of the GA-based path planners. Although Simple GA (SGA) guarantees global optimization, it is much more complicated in real-world path planning problems. For instance, the path planning problem for dual-crane lifting has four hard constraints: node (gene) collision violation, edge (movement between genes) collision violation, node coordination violation and edge coordination violation. The resulting feasible space of the problem is highly non-linear and composed of narrow tunnels, especially for complex environments. This complex feasible space challenges the search ability of GA and any other general path planning methods. Prior path planning algorithms usually transfer constraints to safety penalties and incorporate them into evaluation functions as weighted sums. However, linear combinations cannot effectively ensure GA to eliminate violations of all hard constraints. Instead, it is important for these problems to have the GA monitoring different stages of evolution and adjusting the selection pressures and reproduction strategies accordingly.

1.3 Objectives

The goal of this research is to solve the problems mentioned above with four objectives described below:

(i) To develop a novel collision detection algorithm for lifting path planning purposes. This collision detection algorithm should efficiently handle the complex environments represented as meshes and point clouds. It should also be able to perform efficient continuous collision detection for the cranes and lifting targets.

(ii) To provide formulations of the optimization problems of single-crane and dual-crane lifting path planning. Lifting paths, costs, and safety factors are to be mathematically modeled as solutions, objectives, and constraints. These formulations should also be suitable for GA's representations and implementations.

(iii) To design effective and efficient planners for the single-crane and dual-crane lifting. The path planners should produce highly optimized lifting paths. In the meantime, they also need to be sound in handling motion non-linearity and uncertainties, effective in searching within narrow spaces and efficient in processing complex environments.

(iv) To develop a lift planner cum crane simulator system to help improve the safety and productivity in crane lifting operations. The system should be able to make use of accurate geometric data of real industrial plants to perform automatic lifting path planning. To achieve fast interactions and plannings, it also requires the system to unify various types of geometric data and implement the collision detection and path planning algorithms efficiently.

1.4 Contributions

To achieve the above-mentioned objectives, this research has developed several novel algorithms and techniques. In summary, the contributions of this study include:

(i) An MDM representation of complex industrial plants and sites is proposed to enable efficient collision detection. Based on this representation, image-space collision detection is conducted. This image-space approach provides a basis for the efficiency of the GA-based path planners. For single-crane lifting, this representation is combined with Analytical Swept Volumes (ASVs) for efficient continuous collision detection. That is, the space swept by the lifting target when moving from one gene (configuration) to the next is defined analytically. These ASVs are constructed with information acquired through forward kinematics. For dual-crane lifting, the Triangle Swept Spheres (TSSs) are used as

swept volumes of the lifting target to be checked with the MDMs. During collision detection of candidate paths in the population, the proposed image-space collision detection enables three levels of parallelization: pixel level, chromosome level, and population level. It deals with chromosomes in the population, genes in the chromosomes and pixels in the MDM simultaneously with parallel threads. This type of image-space collision detection has not been applied before for path planning of articulated robots.

(ii) Comprehensive mathematical formulations are proposed for both single-crane and dual-crane lifting path planning which are highly suitable for GA's representations. The path planning problems for single-crane and dual-crane lifting are formulated as multi-objective nonlinear optimization problems with implicit constraints. To fit into the representations of GA, the lifting paths are defined as linear chromosomes with each gene containing a set of values of the manipulation variables. As a consequence, a lifting path comprises a set of nodes which are key-frame configurations and a set of edges which are the local paths between neighboring nodes. The evaluation functions are formulated as piecewise continuous functions which enable GA to control the selection pressures for paths in different stages. Constraints are formulated based on analysis of the kinematic and physical structures of the single-crane and dual-crane lifting systems. Particularly, in the dual-crane lifting problem, the closed kinematic chain is decomposed into two sub-systems: a manipulation sub-system regarding the two cranes which can be solved through forward kinematics and a suspended sub-system regarding the lifting target whose state can be approximated by information acquired from forward kinematics.

(iii) MSPGA-based path planners (for both single-crane and dual-crane lifting) are developed to achieve high success rates and solution qualities in complex and narrow free spaces. The planners exploit the Lexicographical Goal Programming (LGP) strategy used in general multi-objective optimization. Based on this strategy, novel adaptive plans are developed to improve the search ability of GA in the narrow and complex feasible spaces. This customization of the

MSPGAs is innovative which significantly contributes to the problem solving in this research. In the planners, the multiple hard constraints and objectives are embedded in the fitness functions and crossover operators with predefined priorities. The fitness functions are designed in piece-wise continuous forms which reflect multiple stages of optimization according to the priorities of objectives and penalties. High priority objectives or constraints are optimized ahead of low priority ones. The crossover operators are designed in a way that candidates violating higher priority constraints are more likely to be filtered. With the help of the prioritized fitness functions and crossover operators, the multiple constraints are solved from high to low priority during the evolutionary iterations and finally produce feasible solutions. Moreover, the research also proposes adaptive mutation operators to perform local disturbance and smoothing of the candidate paths. In the dual-crane scenario, a coordination mutation operator is also introduced to help supply well-coordinated genes for the population.

(iv) A hybrid C-space collision detection strategy is proposed to achieve optimal computational performances of the MSPGA-based path planners. In this strategy, collision detection information of the manipulation components of the cranes is coded into 2D C-spaces. Each entry in the C-spaces represents the validity of a certain swinging and luffing angle of a crane. Collision detection of the lifting target is computed online using updated ASVs and TSSs during GA iterations. This novel collision detection strategy provides a trade-off between the pre-processing and query time for the MSPGA-based path planners. The use of hybrid C-spaces, especially to handle the manipulation sub-system of dual-crane lifting, is new to the best of our knowledge.

(v) TSS based swept volumes are developed and utilized to handle effectively and efficiently the dual-crane lifting target. The motion uncertainty of the dual-crane lifting target is bounded by introducing distance thresholds from the approximated center lines which are acquired from forward kinematics. These center lines and thresholds are then used to construct the TSSs. The TSSs serve as swept volumes of the lifting target between neighboring genes in the dual-crane

path planner. This is the first time that TSSs are proposed and used to bound the motion between consecutive path steps.

(vi) A prototype system for the lift planner cum crane simulator is developed to serve both training and planning purposes. By integrating the technologies developed in this research, the system incorporates many functions including simulation, proximity warning, safety monitoring, lifting path generation and validation functionalities. It supports and processes various types of geometric data to model industrial environments efficiently and accurately. Data from laser scanning, PDMS, Smartplant and BIM are unified into MDM representations in the system. The parallel collision detection and path planning algorithms are fitted into the hierarchical architecture of CUDA to take advantage of GPU powers. Consequently, the prototype system could provide a novel solution of automatic lift planning as well as intelligent and realistic training.

1.5 Outline of the Thesis

The rest of the thesis is organized as follows:

Chapter 2 reviews the existing technologies and applications of collision detection and robotic path planning. Particularly, in-depth discussions of prior arts on image-space collision detection and lifting path planning closely related to this research are given in this chapter. In order to make the thesis self-contained, a short survey of the basic concepts and designs on GAs are also provided in Chapter 2.

Chapter 3 describes the proposed GPU-based image-space collision detection algorithm designed for simulation and lifting path planning purpose which is targeted for complex environments. Multi-level depth maps are introduced in this chapter. The performance and scalability of the collision detection algorithm are validated and presented in the experiments.

Chapter 4 investigates an MSPGA-based path planning algorithm for single-crane lifting. The mathematical formulation of the single-crane lifting path planning problem is discussed. The design of the GA-based path planner including the fitness function and reproductive operators are introduced. This chapter also presents the ASV-based

continuous collision detection which handles the nonlinear motions of the crane and the lifting target in single-crane lifting. The hybrid C-space strategy which helps to reduce the planning time is also introduced in this chapter. Comparison between the fitness function in [1] and proposed one is provided. Finally, the chapter discusses the influence of parameter designs on the performance of the planner.

Chapter 5 presents the MSPGA-based path planner for dual-crane lifting. This planner also applies the MSPGA framework and the hybrid C-space strategy introduced in Chapter 4. However, contents in Chapter 5 focus on the unique problems encountered in dual-crane lifting path planning which is totally different from the single-crane scenario. This chapter firstly illustrates how the dual-crane lifting system can be decomposed into two sub-systems. Then, the mathematical formulation of the dual-crane lifting path planning problem is developed based on the kinematic and physical analysis of these two sub-systems. This chapter also shows how the dual-crane coordination constraints are formulated and incorporated into the fitness function and the crossover operator using the LGP strategy. Moreover, It introduces the constructions and applications of TSSs to handle the motion nonlinearity and uncertainty of the lifting target. Finally, comparisons to two methods in [1] and [2] are presented to show the effectiveness and efficiency of the proposed algorithm.

In Chapter 6, the design, implementation and applications of the lift planner cum crane simulator are discussed. This chapter shows how the information of real industrial plants and sites can be captured, processed and imported into the system. GPU parallelizations of the collision detection and the MSPGA-based path planning algorithms are also presented in this chapter. An experiment in an industrial plant is provided as a case study to validate the system.

Finally, Chapter 7 concludes the research by highlighting the achievements and discussing the limitations. It also discusses possible future work to address these limitations.

Chapter 2

Literature Review

2.1 Collision Detection

2.1.1 Introduction

Collision detection (CD), sometimes called intersection detection or interference detection, is widely used in many fields. Collision detection is a building block of physical simulations, robotic motion planning and virtual reality and games. The collision detection problem is to determine the interference between a set of objects represented as geometric models. These objects can be either static or dynamic, possibly deforming with time. Outputs of collision detection include boolean result indicating whether two or more objects collide, the proximity between objects and the set of colliding features. Often, for the purpose of calculating collision response, penetration depth is also computed in the CD process. The collision of moving or deforming objects can be handled in either a discrete or continuous way. Discrete Collision Detection (DCD) typically considers the interference between objects at discrete time points, while Continuous Collision Detection (CCD) handles the continuous motion of objects during simulation time steps.

The popular game "Grand Theft Auto" (Figure 2.1) illustrates how heavily is the collision detection process performed in virtual environments. In this game, the character explores big cities represented as polygonal models. The scene contains buildings, street lamps, and trees that comprise the static portion of the environment. Many dynamic obstacles such as walking people, moving cars and bicycles also exist in the scene. Collisions and the subsequent responses when encountering static or

Figure 2.1: Scenes from the game "Grand Theft Auto".

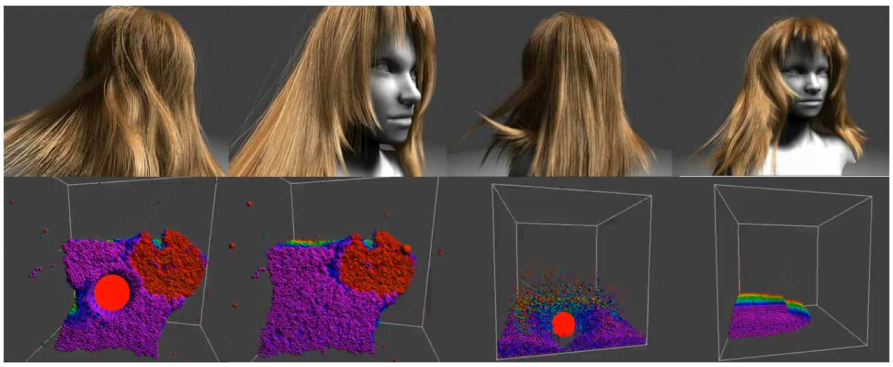


Figure 2.2: Two examples of physical simulation. Upper row: Hair simulation; Lower row: Particle simulation.

dynamic obstacles are required to be computed accurately and interactively, which contributes directly to the realism of the games.

Physical simulations have higher requirements of accuracy than games do. Serious simulations demand accurate collision detection and responses which must be not only visually acceptable but also scientifically accurate. High accuracy CD required in physical simulations is computationally expensive and time-consuming [7]. Examples of physical simulations include hair modeling [8], cloth simulations [9] and particle simulations [10]. Figure 2.2 shows the snapshot captured from Nvidia's simulation demos [11, 12]. In the case of hair simulation, both hair-hair and hair-body interactions need to be handled. The huge amount of hairs in a character result in millions of tessellated triangles, bringing enormous challenges to the collision detection.

Figure 2.3 shows a robot "Robonaut" designed by NASA's Johnson Space Center in Houston, Texas [13]. When the robot is asked to conduct tasks such as grabbing, holding, hand shaking and writing, it needs to recognize the shapes and locations of the targets and obstacles. Interferences between the robot and these obstacles need

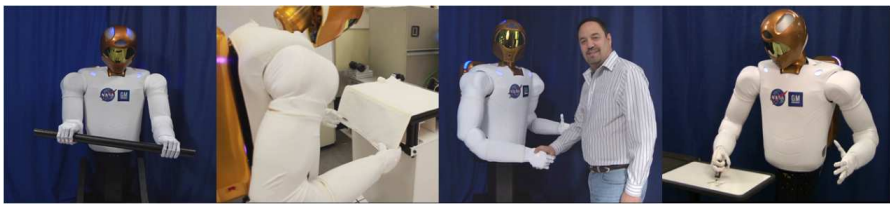


Figure 2.3: The robot "Robonaut" is performing grabbing, holding, hand shaking and writing.

to be handled efficiently in order to quickly plan applicable motion paths. For these robotic motion planning applications, the complex nonlinear movements of the robots pose challenges on the efficiency and accuracy of CD.

2.1.2 Collision detection for polygonal models

Polygonal models are widely used to represent agents, robots and environment objects in virtual reality, physical simulations and robotic motion planning. The models may vary from raw polygon soups containing no topology information to highly structured representations like the winged-edge [14] and half-edge meshes [15]. This section reviews prior studies on collision detection and continuous collision detection for 3D polygonal meshes, especially triangular meshes.

Collision detection for polygonal meshes is conducted by detecting interference between primitives (triangles, edges and vertices). A naive and robust way is to consider all possible pairs of triangles and check interferences between them. This direct approach is valid for a small amount of triangles. However, when shapes become complex, this approach suffers from its $O(n^2)$ complexity. A commonly used strategy to improve the efficiency is to divide the CD and CCD process into two phases: the broad phase and the narrow phase [16].

The narrow phase of collision detection detects the interference within the Potential Colliding Sets (PCS). A classical way to test the interference between a pair of 3D triangles is to consider the 6 elementary Edge-Face (VF) contacts [17]. In this method, co-planar or degenerate cases are handled specially. For CCD cases where triangles are conducting linear continuous motions between successive time steps, 15 elementary tests including 6 Vertex-Face (VF) tests and 9 Edge-Edge (EE) tests are required [18].

In this case, the proximity or collision tests can be handled by two steps: calculating the time when the features (vertices, edges and faces) become co-planar, which is solved with cubic equations, and determining the planar distance between the features at the co-planar time [19]. Another approach for narrow phase checking is to combine the separating-axis test with convex decomposition. The separating-axis test projects the triangles onto 11 separating axes taking into account the direction of the triangle normals and Cartesian products of the edges. If non-intersecting intervals of the triangles appear on any tested axes, the pair of triangles are regarded as separated. On the other hand, if the projected intervals intersect for all the 11 axes, the pair of triangles are surely colliding with each other. Similar concepts have been used to determine the collision between Oriented Bounding Boxes (OBBs) [20] and Axis Aligned Bounding Boxes (AABBs) [21]. In the narrow phase of CCD algorithms, Swept Volumes (SVs) can be used to bound the space swept by primitives during continuous motions.

The major task of the broad phase is to produce the PCS which contains a reduced number of primitives for narrow phase tests through culling distal or non-colliding primitive pairs. Typical solutions include the *Sweep and Prune* methods [22], Spatial Partitioning (SP) [23], and Bounding Volume Hierarchies (BVHs) [24]. In the Sweep and Prune method introduced in [22], objects are projected onto a predefined sweeping axis and coded as an interval $[b, e]$. The b and e values of projected objects can be then sorted in an ascending manner. When sweeping through the axis, an object is marked as active when encountering its b value and deactivated when leaving its e value. Two objects are considered as potentially colliding if they appear in the active object list at the same time. This technology can be naturally combined with the AABBs by regarding them as three-dimensional intervals. The sorted list of intervals in the three coordinate axes is maintained in order to prune non-colliding AABB pairs. This method has been further developed into the I-COLLIDE system [25] which deals with exact distance computation for multiple convex bodies and the V-COLLIDE system [26] which aims to output binary object-level collision results for arbitrary-shaped triangular models. These sweep and prune methods can achieve good performance when the objects are evenly distributed, but the sorting performance may become $O(n^2)$ when objects are clustered [17].

Octrees [17] are one example of spatial partitioning hierarchies. The initial purpose of designing octrees is to efficiently represent graphic data [27]. The use of octrees in collision detection is introduced by Shaffer [23]. Differed from uniform spatial partitioning, octrees exploit the idea of Level of Details (LOD) by dividing the space into eight sub-spaces uniformly and recursively until meeting the termination criteria. For example, the construction can be terminated when the numbers of primitives contained in all leafs are smaller than a certain preset number. By maintaining an octree, only objects or primitives in the same or neighboring nodes are required to be tested. Octrees have been applied later to construct an Octree Distance Map (ODM) in [28] for assisting motion planning. In their algorithm, pre-computed and float-coded distances to obstacle nodes are stored in each node of the octree for later look-up in the planning process.

Other spatial partitioning hierarchies include k-D trees and Binary Spatial Partitioning (BSP) trees. A k-D tree recursively partitions the k-dimensional space into half-spaces using hyperplanes whose normals are aligned with the coordinate axes. Herzen et al. [29] have applied the k-D trees in subdividing parametric spaces and have observed good culling performance on the time-dependent parametric surfaces. The BSP trees can be regarded as a generation of the k-D trees where the partitioning hyperplanes are allowed to have arbitrary orientations. Work of [30] has illustrated how a self-customizing BSP tree can offer significant speedups by making use of temporal coherence. The SP hierarchies can help to quickly localize the CD tests in static or almost static situations. However, for dynamic environments, the mentioned SP hierarchies suffer from the frequent reconstructions of the trees which can be quite costly for complex environments.

Many studies make use of BVHs to perform the broad-phase culling. The Bounding Volumes (BVs) are typically chosen as simple shapes such as spheres, AABBs, OBBs, k-dops and convex hulls that are easy to represent and perform collision tests. These hierarchies may contain a single type of BVs or combine multiple types of BVs. The more complex the BVs are, the more time-consuming is the construction of the BVHs. The trade-off between simplicity and tightness of the BVs is important but often problem-specific. For far-separating objects, simple BVs such as spheres and AABBs

bring superior performance. However, for scenes with many closely distanced objects, tighter BVs are more suitable [17]. [31] have shown how a BVH-based CCD algorithm outperforms a BSP tree based one by using 4D bounding boxes for broad-phase culling and sphere trees for approximate narrow-phase collision tests. A benchmark RAPID system based on OBB hierarchies has been introduced and tested against AABBs by [20]. The results have shown that the OBB trees perform specifically well in close proximity situations. Recently, the OBB trees have been utilized by [32] to calculate continuous penetration depth between two objects. Since the objects are already intersecting each other, the OBB trees can help to quickly localize the penetration position. The potential benefits of using AABBs trees for complex and deforming objects has been shown in the work of [21, 33] by comparing its performance with OBB trees on several benchmark cases. This feature of AABB trees has been further exploited in [34] where dynamic link-level AABB trees are constructed using Taylor models and used to perform spatial culling for an articulated robot. This algorithm uses Conservative Advancement (CA) for exact contact determination and investigates temporal culling to further reduce the complexity of CCD. The k-dops (DOP for"Discrete Orientation Polytope") introduced by [35] targeted especially at dynamic objects and complex environments. A k-dop is a convex polytope whose faces have orientations chosen from a fixed set of directions of size k. The performance is comparable to the RAPID system [20] with its parameters carefully designed. For the purpose of CCD in [19], the k-dop trees are used to bound the SVs of triangles and perform spatial culling. Lately, a BVH using k-dops based on clustering of triangles has been developed in [36] to handle the continuous self-collision detection of complex skeleton models. This method has improved the classical k-dop trees by constructing a tree for each triangle cluster corresponding to observation points and have shown comparable performance with the ICCD method using connectivity-based culling [37].

A special family of BVs is the Sphere Swept Volumes (SSVs) introduced in [38]. The family of SSVs includes the Point Swept Spheres (PSSs) that are represented as sets of spheres, the Line Swept Spheres (LSSs) that have capsule-like shapes and the Rectangle Swept Spheres (RSSs) which are the shape swept by spheres along 3D rectangles. Interference checking of the SSVs is particularly simple since it can

be reduced to the proximity computation between the core primitives (points, line segments, and rectangles) and the obstacles. Larsen et al. [38] have illustrated the use of SSV based BVHs containing single or multiple types of SSVs for broad-phase culling. External Voronoi regions are utilized to perform the narrow-phase proximity tests for the core rectangles. Work of [39] has illustrated how LSSs can be used as narrow-phase SVs by combining them with dynamic AABB trees and OBB trees for broad-phase culling. In the field of robotics, [40] has applied LSSs to bound the motion of the skeleton links of robotic manipulators. A dynamic BVH consisting of LSSs is constructed for the manipulator with the radii of the LSSs taking into account the velocity bounds of the links. The results of [40] have indicated that the LSSs can offer much tighter bound than spheres for the tested manipulator. Tight-fitting RSSs are applied by [41] to conduct proximity calculations. Their algorithm has achieved fast performance with the help of GPUs.

2.1.3 Parallel collision detection

The above-mentioned acceleration data structures can speed up CD and CCD applications significantly. However, when the objects and environments get large and complex, it becomes more and more challenging to achieve real-time performance. This limitation leads to the popularization of parallel collision detection algorithms [7].

2.1.3.1 Collision detection on multi-core CPUs

The multi-core CPU based methods usually decompose the traversal, updating or checking of BVH nodes into parallel tasks. The load balancing of the threads is attempted by estimating the expected number of further expansions brought by expanding the considered node or node pair. Thomaszewski et al. [9] have introduced a dynamic task-parallel approach for multi-core collision detection. Their algorithm takes advantage of temporal coherence by keeping track of the number of children for each node in the BVH. Tasks are then dynamically assigned to parallel threads from the shared task pool according to the number of children to be expanded. Similar

to the idea in [9], Kim et al. [42] have made use of multi-core CPUs to perform BVH updates and traversals. Their algorithm decomposes the traversal of BVHs into inter-CD task units in a way such that each of them accesses a different set of nodes. These task units are then assigned to parallel CPU threads for processing. For CCD of deformable objects, Tang et al. [43] has developed a MCCD system based on multi-core CPUs. In their algorithm, load balancing is achieved by constructing a Bounding Volume Traversal Tree (BVTT) and maintaining a BVTT front which records the number of nodes that have been expanded at the locations where the traversal of the branch terminated. A Front Based Decomposition (FBD) is then performed to assign the fine-grained tasks to threads. The performance speed-ups obtained by these multi-core CPU based algorithms are heavily dependent on how well balanced the loads are on parallel threads. The load balancing is essentially influenced by the accuracy of the load estimation hypothesis. These methods can perform well occasionally if temporal coherence could be exploited. In other cases, they are easy to be trapped in bottlenecks because of the difficulties to estimate the load distributions on BVHs.

2.1.3.2 Image-space collision detection

An alternative approach is the image-space collision detection which avoids the time-consuming pre-processing such as constructing BVHs [44]. Image-space collision detection usually takes advantage of the hardware accelerated graphic libraries such as OpenGL and DirectX. The objects are rendered, and information such as depth and visibility values are coded and stored in the GPU depth buffers, stencil buffers and color buffers. Interference tests on pair-wise or multiple objects can thus be conducted on the rendered images.

The idea of image-space collision detection has been introduced in [45]. The algorithm proposed in [45] rasterized objects by sweeping scan lines. When an object surface is encountered, their algorithm stores a depth value and a link to the original object in the pixel. Intersection tests are then conducted on the sorted z-list on each pixel. Collisions of objects are reported if overlaps of z-intervals are found. Later, a hardware-accelerated image-space collision detection algorithm called REndered COllision DEtection (RECODE) has been introduced by [46]. Their algorithm renders

pair-wise convex objects on the Minimum Overlapping Region (MOR). Collisions are determined by comparing the min-max values on the depth maps. Their results show that the image-space system can outperform the I-COLLIDE system which uses SAT and OBB trees to perform object-space collision detection in complex scenes. Heidelberger et al. [47] have introduced the use of Layered Depth Images (LDIs) as the representation of the intersection volumes. In their algorithm, a Volume of Interest (VOI) is firstly acquired through performing pairwise AABB intersection tests. Then the two objects are rendered in directions restricted by the VOI. Two LDIs are thus produced for each object with entry points and leaving points of the object recorded. The actual collision detection is conducted by performing intersection tests on the entry-leave pairs. This work has been extended into [48] by adding on face orientations in order to handle self-collisions. Research in [49] has further investigated the LDIs by rasterizing the objects in the PCS on three orthogonal directions. The LDIs are then used to analyze the repulsion forces on penetrating surfaces. Another group of studies perform broad-phase culling using image-space approaches. The CULLIDE system proposed in [50] employs OpenGL occlusion queries to obtain the PCS in the broad phase. Serial exact primitive tests are conducted in the narrow phase. The method has been further developed into the Quick-CULLIDE system [51] where intra-object collision are handled by generalizing the formulation of PCS. The idea has also been adapted to a CCD algorithm by [16]. In this case, the visibility queries are conducted to cull non-colliding SVs of primitives. These image-space methods based on graphic libraries are easy to implement, and their performances have been verified. However, the first class of these algorithms performing narrow phase checks in the image space requires reading back the images from the graphical hardware. These read-backs, however, should be avoided due to the unsymmetrical design of the memory bandwidth between the CPU and GPU. The second class using occlusion queries to perform broad-phase culling can be highly dependent on the choice of projection directions. This restriction is similar to the sweep and prune methods.

2.1.3.3 GPU-based collision detection

GPUs are equipped with tremendous computational horsepower and high memory bandwidths. They can bring significant speed-ups to various applications [52]. General

Purpose GPUs (GPGPUs) are new generation GPUs aiming at handling more general, complex and computational-intensive processing. GPGPUs provide a complete functional set of operations working on arbitrary length data. A GPGPU contains several Streaming Multiprocessors (SMs) which can run hundreds of threads concurrently. The SMs are equipped with caches and control units shared by internal threads.

CUDA C [52] is a typical GPU accessing API designed by nVIDIA as an extension of the standard C language. It allows programmers to allocate GPU memories and run kernels on parallel threads in a C/C++ style [52, 53]. Various accesses to GPU memories are provided in CUDA C/C++. Global memory, constant memory and texture memory lie in the global physical memory while shared memory resides inside SMs. Local memory and register memory are only usable for the threads who have allocated them. CUDA has a hierarchical thread structure which reflects the hierarchical hardware architecture. Each launched kernel is handled by one thread grid. The thread grid consists of an array or matrix of thread blocks and the blocks contain similar matrices of threads.

The GPGPU architecture has been applied to speed up collision detection. Most of them use GPUs to accelerate object-space algorithms. These investigations include those using GPUs to perform parallel narrow-phase primitive tests. [39] has investigated the use of GPUs on CCD of articulated robots. Their algorithm uses LSSs as the SVs of the links and uses GPU to check the interference between the LSSs and the environment in parallel. The HPCCD (short for Hybrid Parallel Continuous Collision Detection) system introduced in [42] uses GPUs to perform elementary tests for potentially intersecting triangles remained from the BVH-based culling. Other algorithms apply GPUs in both phases. The research in [54] has shown how GPUs can be used both to traverse AABB trees and conduct primitive tests through occlusion queries. Their algorithm applies breath-first traversal on the AABB trees by constructing a 2D node pair index map. Task decomposition of the node pair index map is obtained by maintaining an overlap count map accumulating the number of expanded nodes. The method in [41] conducts parallel BVH construction, updating and traversal in GPUs to perform proximity computation. Their algorithm also uses GPUs to conduct the complex elementary tests between the tight-fitting RSSs and OBBs. Recently, GPUs

have been utilized to perform parallel collision detection for motion planning of high DOF robots [55]. GPU parallelization is applied in both the traversal of BVTTs and the collision test of the BVs. Their algorithm uses a clustering scheme to improve the coherence of the parallel traversal of BVTTs. The advantage of using GPUs against multi-core CPUs have been proven by previous studies. For example, the GPU implementation of the k-D tree based algorithm in [56] have shown significant speedup compared to that on multi-core CPUs. The investigation in [57] has also demonstrated that the GPU-based object-space algorithm with deferred front tracking outperforms their previous one [43] using multi-core CPUs.

2.1.4 Discussions

Serial object-space collision detection algorithms handle simple scenes efficiently by applying BVHs and SP hierarchies. However, the construction, updating and traversal of these acceleration data structures become costly in complex environments. Parallel collision detection speeds up these algorithms with multi-core CPUs and GPUs. However, the task decomposition and load balancing for these complex tree structures might be difficult for many occasions. Therefore, this research investigates GPU-based image-space collision detection and optimizes the procedures and data structures for the hierarchical CUDA architecture.

2.2 Genetic Algorithms

GAs are a set of evolutionary search algorithms targeting at global optimization. GAs are widely used in complex optimization problems such as scheduling and path planning. They offer a basis for the path planning algorithms proposed in this thesis. This section introduces fundamental ideas, designs and variations of GA.

2.2.1 Basic theory

The idea of GA has been first put forward by John Henry Holland in the 1960s. In the book "Adaptation in Natural and Artificial Systems", Holland has described GA as a computational abstraction of Darwinian biological evolution [58, 59]. He has introduced a simple mathematical model for GAs consisting of the following components:

(i) $\alpha = A_1, A_2, A_3, \ldots$: The set of possible solutions.

(ii) $\Omega = \omega_1, \omega_2, \omega_3, \ldots$: The set of adaptive operators which maps into a probability distribution over α.

(iii) I: Set of inputs from the environment to the adaptive system.

(iv) $\tau = \alpha \times I \rightarrow \Omega$: The adaptive plan doing operation selection based on the inputs and the current solution set.

The adaptive plan is conducted in discrete time steps. With the time step parameter denoted as t, Holland has proposed the abstraction of the GA process as:

$$\tau(\alpha(t), I(t)) = \omega_t(\alpha(t)) \in \Omega, \omega_t(\alpha(t)) = \delta(t+1) \tag{2.1}$$

Where $\delta(t+1)$ denotes the probability distribution over α according to which $\alpha(t)$ is sampled and produces $\alpha(t+1)$.

He has then illustrated adaptive optimization plans in six different domains: genetics, economics, game playing, pattern recognition, control and function optimization and central nervous systems. As a conclusion, he has suggested a similar set of parameters for all the problems:

(i) α: The domain of possible solutions represented in problem-specific ways.

(ii) Ω: The reproductive operators.

(iii) T: The reproductive plans scheduling application of operators.

(iv) ϵ: The evaluation functions for solutions.

(v) X: A ranking of plans in T.

A formal format of GA can be given with the following standard components:

(i) Chromosomes: Genetic representation of solutions. The chromosomes can either have fixed or variable length

Table 2.1: Crossover strategies.

Strategies	**Descriptions**
One point crossover [60]	Split each chromosome into two sub-ones and exchange between parents
Two point crossover [61]	Split into three sub-ones and exchange the middle portions
Uniform crossover [62]	For each bit or element in the chromosome, randomly choose to inherit from its mother or father
Arithmetic crossover [63]	Conduct arithmetic calculations between parent genes to obtain the child gene

(ii) Genes: Basic components of a chromosome. The genes can be organized in different ways such as linear arrays, trees and so on.

(iii) Population: The set of chromosomes. A GA may contain one or multiple populations.

(iv) Generations: Discrete time steps or the populations associated with time steps.

(v) Crossover: Recombination operators.

(vi) Mutation: Reproduction operators which alter the chromosomes randomly.

(vii) Fitness function: Evaluation criteria for chromosomes.

(viii) Selection: Selecting chromosomes to perform reproductions.

Among those notations, selection, crossover and mutation are usually called genetic operators. There are many designs for each genetic operator. Variations of the crossover operator include one-point, two-point, multi-point, parameter-based strategies and so on. For mutation, common strategies include single-point, multi-point and parameter-based mutations. Both the crossover and mutation operators are conducted under certain possibilities which may vary adaptively during the evolution process.

Table 2.1 and Figure 2.4 show some common crossover operators. The simplest crossover strategy is selecting one of the parents according to certain criteria and directly copying it into the off-spring. This strategy may cause a total loss of the

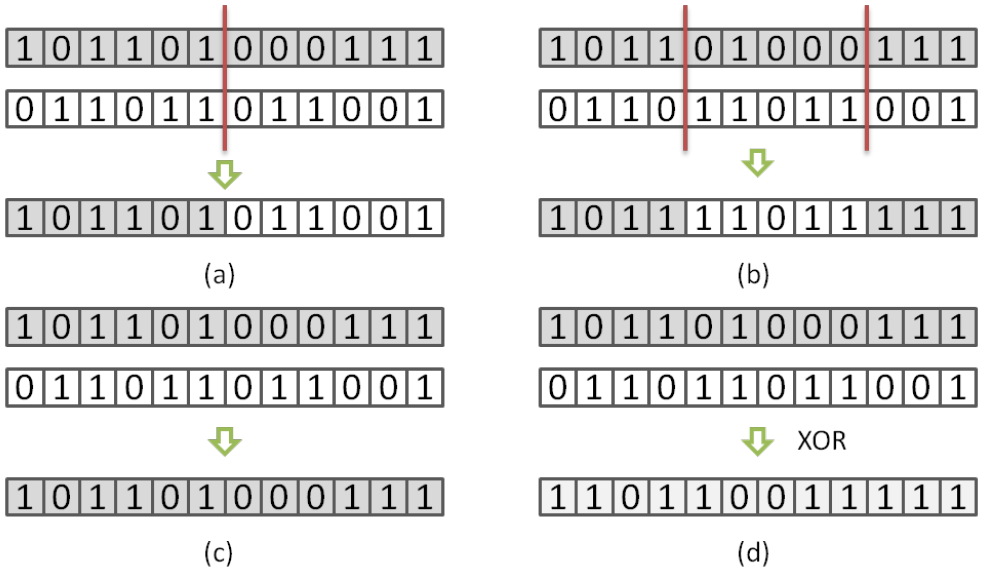


Figure 2.4: Different crossover strategies: (a) One-point crossover; (b) Two-point crossover; (c) Direct selection from parents; (d) Parameter based crossover.

information from one of the parents and thus is not commonly used. One-point or multi-point crossover exchange sub-strings from parents and combine them into new offspring. In parameter based crossover or bitwise crossover, bitwise arithmetic calculations on parent genes are conducted for each gene in offspring. When dealing with specific problems, the crossover strategy should be carefully chosen to ensure that the operator reserves and combines important features from parents.

GAs have been used in solving the Traveling Salesman Problem(TSP) [64] which considers the shortest path problem in a graph of cities to be traveled by a salesman. In a TSP problem, solutions are represented as permutations of IDs indicating different cities. The crossover strategies involved in the TSP problem are called "permutation crossover" operators. There are mainly three typical types of permutation crossover operators: Partial Mapped Crossover (PMX), Ordered Crossover (OX) and Cycle Crossover (CX) [65]. Figure 2.5 illustrates the mechanisms of PMX [65] and OX discussed in Hamidinia's work [66].

Instead of taking two parents to produce offspring, mutation operators directly alter independent chromosomes. Figure 2.6 shows two typical mutation strategies: single-point and multi-point mutations [59].

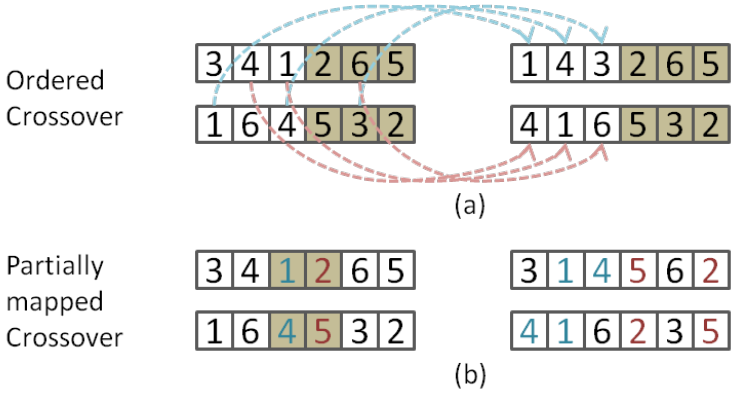


Figure 2.5: Two permutation crossover operators: (a) Ordered crossover; (b) Partially mapped crossover.

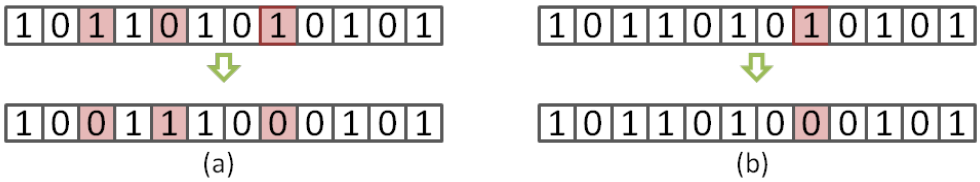


Figure 2.6: Two mutation strategies: (a) One point "flip bit" mutation; (b) Two point "flip bit" mutation.

The process of applying fitness function on chromosomes (strings) is called fitness evaluation. Fitness functions are highly problem-specific. However, there are some common classifications such as adaptive (improving with time) and non-adaptive, scaled (by scaling functions) and non-scaled. Among various fitness functions, several of them are selected as benchmarks for testing GA designs as general function optimizers [62].

2.2.2 Simple genetic algorithm

A SGA is a GA with the following features [60, 67]:

(i) The genes are binary bits

(ii) Use linear chromosomes with a fixed length

(iii) Follow the simple GA procedure (Figure 2.7)

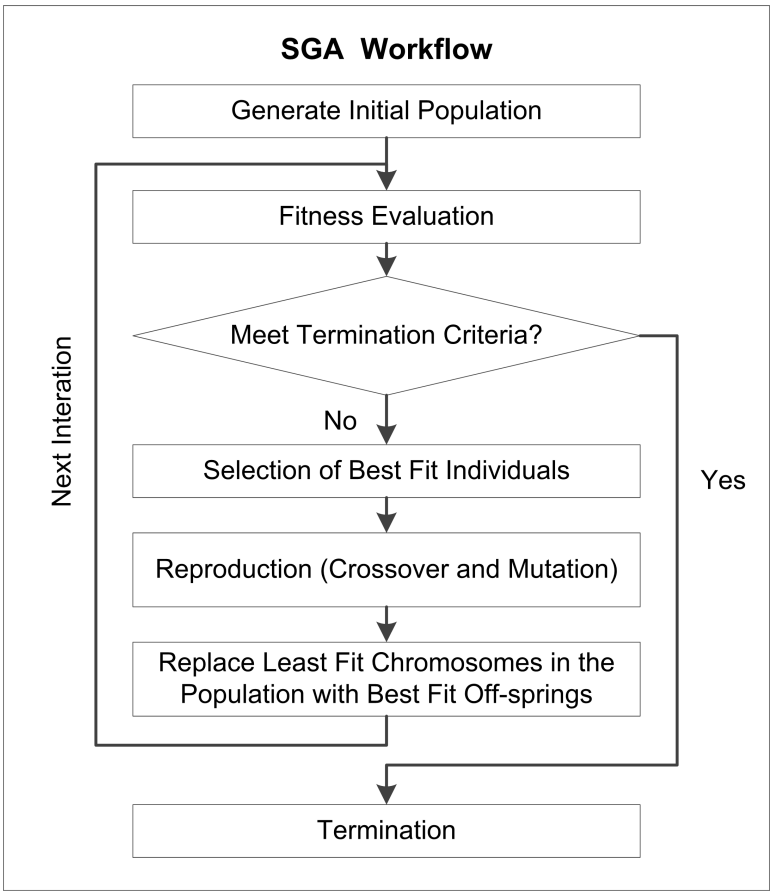


Figure 2.7: Standard procedure of simple genetic algorithm.

There exist a variety of selection strategies. Typical selection criteria are listed in Table 2.2.

When all the reproduction and selection strategies are determined, an additional step for designing SGA is to decide the termination criteria. Proper termination criteria can save processing time of GA in the sense of preventing unnecessary iteration runs after the desired goal has been achieved. Common termination criteria [73, 74] are listed below:

(i) When achieving a certain number of improvements;

(ii) When the average quality of the population has not been improved for a certain number of generations;

(iii) When reaching some predefined fitness value;

Table 2.2: Commonly used selection strategies and their descriptions.

Strategies	**Descriptions**
Roulette wheel selection	Probability of each chromosome to be selected is proportional to its fitness value
Rank selection [68, 69]	Probability of chromosomes to be chosen is proportional to the rank
Steady-state selection [70]	Only select a few of chromosomes to produce offspring. Replace the worst ones with the new chromosomes. Most chromosomes survive to the next generation
Elitism [71]	Copying the best chromosomes directly into the next generation.
Tournament selection [60]	Selecting crossover parents through tournaments between a randomly selected fixed-size group
Reward-based selection [72]	Chance of strings being selected for crossover is proportional to the fitness that the individual have accumulated and inherited from its parents

(iv) When all the individuals are identical (in genetic drift cases);

The first strategy is the simplest and most commonly used one. The second one may easily lead to termination in local minima. The third one can be more effective, but since the fitness value of global optimum can not be predicted in most occasions, the design of the termination fitness is tricky.

New genetic operators like migration, inversion, fitness sharing and mating restriction may also be used for some particular cases. Migration is a part of multi-deme parallel GA, which helps the sub-populations to escape from local optima. Inversions are mainly used in messy GAs. The fitness sharing [75] operation can alternatively be added to the GA scheme to encourage niching in landscape peaks. Mating restrictions [76] are required for scenarios when mating between strings with large difference brings lethal to the population.

2.2.3 Parallel genetic algorithms

Most execution time of GA is usually taken by the fitness evaluation process. This process if performed on a population with fixed size and consisting of uniform-structured

chromosomes, however, can be parallelized by decomposing the tasks of evaluating independent strings. Other operations like crossover and mutations also have similar parallel features at the chromosome level. Thus, SGAs are naturally suitable for parallelization.

This feature of GAs leads to the design of *Master Slave Parallel GAs* (MSPGAs) which are parallel versions of serial SGAs. In an MSPGA, the host processor is responsible for the flow control and tasks are decomposed and assigned to the slave processors to be performed parallelly. Ismail and Ali [77] have implemented MSPGA with MPI and have observed a prominent speedup due to parallelization. Zhao and Man [78] have applied MSPGA in optimizing reactive powers in power systems and have reported better results than other approaches.

Another type of parallel GAs are called *multi-population GAs* or *multi-deme GAs* [79, 80]. This type of PGA divides the population into several sub-populations and makes each sub-population evolve independently. Multi-population PGA introduces a new operator into the GA family called *migration*. The main idea of migration is to exchange individuals between connected sub-populations with some possibility for each given or adaptively chosen time step. The effect of the migration operators and the efficiency of multi-deme PGA have been investigated by many researchers. It has been shown that the migration operators can break the trap at local optima and bring new driving force to the convergence of sub-populations. With careful parameter designs, multi-deme PGA can obtain population quality as good as serial GA with a faster speed.

Apart from the course-grained multi-deme PGA, fine-grained GAs are also a popular category of PGAs. A fine-grained PGA assigns each parallel processor a single chromosome and constrains the communication within neighborhoods. Fine-grained PGA shows a slower propagation of new schemas through the population due to the lack of global accessibility. However, a significant decrease in execution time can also be expected. Fine-grained PGAs may compromise solution qualities, but have good computational performances [81, 82].

2.2.4 Discussions

GAs have superior optimization capabilities even for difficult optimization problems. Parallel GAs improve the efficiency of traditional GAs by parallelizing them in various ways. Among these algorithms, multi-deme PGAs exploit coarse-grained parallelizations. Although they have improved the search ability of standard GAs, the computational efficiency has been sacrificed. Fine-grained PGAs are highly suitable for massive parallelizations. However, the convergences of fine-grained PGAs are slow due to the lack of global communications in the population. Therefore, MSPGAs which maintain the global optimality of SGAs are chosen and customized to perform path planning in this research. When massively parallelized in GPUs, MSPGAs can produce highly optimized solutions with excellent efficiency.

2.3 Robotic Path Planning

The concept of motion planning comes from the classic *piano mover's problem* [3] which requires a piano to be moved between rooms without hitting the ground, stairs or furniture. Motion planning takes inputs like the geometric representations of the agents and environments and outputs paths or trajectories which the agents can follow to travel from the initial states to the desired goal states.

In robotic motion planning, the collection of all configurations (a complete set of independent parameters in the robot) is called the *configuration space* (C-space in short) of the robot. A *path* $P = c_i$ is a collection of configuration in the C-space of the robot while a *trajectory* also encodes dynamic information such as velocities and angular velocities. The term *path planning* represents queries aiming to generate paths without considerations of dynamics and a *trajectory planning* problem is to output trajectories with dynamic information. This section reviews existing path planning algorithms that either produce feasible paths meeting hard constraints such as collision avoidance and DOF limits or optimal paths evaluated under optimality criteria like motion, time and energy costs.

2.3.1 Discrete search based path planning

Early path planning algorithms adopt concepts from graph search algorithms and perform incremental searches on discretized workspaces. A classic example of these spaces is the map of cities with transportation costs between cities precisely known. In this case, the path planning problem aims to find the shortest path in the graph for traveling from a start node to an end node. The workspaces can also be represented as grids or lattices. A uniform grid or lattice in 2D is equivalent to a graph where each internal node is connected to four or eight neighboring nodes.

Dijkstra's algorithm [83] is a classical algorithm for optimal path planning in graphs or grids. This method propagates from the start node to the goal node by expanding already explored nodes. The algorithm maintains a list of nodes already explored (set A), a list of candidate nodes for expansion (set B) and a list of rejected or unexplored nodes (set C). For each iteration, the node from set B with minimum traveling distance from the start node is chosen to be expanded, and its neighbors are added to set B. The original node is then moved to set A. Dijkstra's algorithm tries to minimize the cost-to-come (cost to travel from the start node to the current position) for expanded nodes and is guaranteed to find optimal paths. Unlike Dijkstra's algorithm which explores the workspace uniformly, the A* algorithm (A* in short) [84] tries to use heuristics (domain knowledge) to guide the search and optimizes the number of node expansions. A* maintains an open list and a closed list. The open list is sorted according to a heuristic function $f(n) = g(n) + h(n)$ where g(n) defines the cost-to-come and h(n) represents a lower bound estimation of the cost-to-go (cost to reach the goal node from the current node). By defining an ideal $h(n)$, the ideal A* defined by [84] has been proven to expand fewer nodes than other admissible heuristic searches. However, when $h(n)$ overestimates the cost-to-go, A* might be trapped into local optimums.

[85] has conducted a comparison on the optimization performance of A* and the Dijkstra's algorithm. The two algorithms and a GA have been used to conduct multi-objective planning tasks using the weighted sum of the objective values to guide the search in a 2D grid represented environment. The results have shown that A* and the Dijkstra's algorithm have similar performance in finding short paths but can be easily trapped into local optima for other objectives. In this case, the Dijkstra's

algorithms can provide better solutions. Kala et al. [86] have applied A*, together with a GA and an Artificial Neural Network (ANN), in path planning within a 2D grid containing moving objects. With re-planning conducted after each step of motion, A* has generated near optimal paths for a point-shaped robot. The algorithm in [87] has applied A* for high-level path planning in a probability based map. Each node in the map contains a possibility of collision estimated using the number of objects in the node. A* is then performed to optimize the cost function which measures the motion distance and collision probability of the path. The detailed low-level path determination is conducted using fuzzy rules considering non-holonomic constraints. Their algorithm has improved the path smoothness of A* and its scalability to higher resolution maps.

To handle environments that are unknown, partially-known or those containing dynamic objects, the A* algorithm has been extended to D* [88], Lifelong Planning A* (LPA*) [89] and D* Lite [90]. D* [88] also maintains an open list like A*. The search starts from the goal position. D* captures the overestimation and underestimation of the cost-to-go for newly explored nodes. These inconsistencies of the cost may be brought by newly found shortcuts or obstacles. When the node about to be expanded has reduced the cost-to-go due to the state update of the map, its neighbors are added to the open list, and the lower costs are propagated to them. For nodes whose cost-to-go have increased after updating the map, D* tends to find alternative shorter ways through its neighbors. If the cost is not able to be recovered, the increase will be propagated to the neighbors. For some cases, closed nodes are also required to be reopened in D*. The LPA* algorithm (LPA* in short) [89] aims to deal with static environments which are not precisely known. Similar to A*, it uses the estimated length of the full path (by summing up the cost-to-come and estimated cost-to-go) to sort the open list. However, LPA* also monitors the consistency of the cost-to-come for newly explored nodes. It only opens inconsistent nodes along the original path. If the old cost-to-come of a node obtained in the initial planning state is larger than the updated value, the cost is updated, and the successors of the node are added to the open list. For nodes whose updated cost-to-come have increased from the old values, the node itself remains in the open list to be expanded in the future. Instead, LPA*

reopens its successors and recalculates the paths for them. LPA* is not able to handle changing environments at traveling since it does not take care of changes in unexplored areas. The D* Lite algorithm (D* Lite in short) is developed from LPA* to handle re-planning in dynamic environments. In D* Lite, if a node u causes increased key value in the path, its expansion is denied. In other occasions, if the cost-to-go has decreased for u, its predictors are updated and inserted into the open list. Otherwise, D* Lite opens u and its predictors and calculates shorter alternative paths for them. Different from D* who sorts the open list using cost-to-go alone, D* Lite optimizes the accumulated length of the whole path.

2.3.2 Sampling-based path planning

Sampling-based path planning algorithms build trees or roadmaps by generating dense incremental sampling sequences. The tree-based searches are performed similarly as the discrete searches except for the new nodes are randomly sampled points instead of unexplored grid or graph neighbors. Sampling-based methods are especially suitable for high-DOF problems since fewer vertices or nodes are used. Popular sampling-based methods include randomized potential fields [91], Probabilistic Roadmaps (PRMs) [92] and Rapidly-exploring Random Trees (RRTs) [93].

The randomized potential field algorithm [91, 94] is one of the earliest sampling-based methods. It improves the original potential field methods which have been applied in guiding the navigation of vehicles [95] and robots [96] in 2D spaces. This algorithm assigns a potential function that combines the attractive forces to the goal and repulsion from the obstacles for each node in the grid. Expansions of the tree try to minimize the potential function that is equivalent to the cost-to-go in many cases. Therefore, the randomized potential field method propagates in a greedy and depth-first manner. When the tree is trapped in local optima, a random walk process is conducted for escaping. The algorithm in [97] uses a heuristic search instead of random walk to help the randomized potential field jump out of local optima.

The PRM method (PRM in short) [92, 98] targets at multi-query problems. PRM constructs a roadmap that can be easily reached from any start or end locations in the pre-processing stage. In this construction process, collision-free sampling points are

inserted into the roadmap and be connected with its nearest neighbors in the roadmap using local planners. Sampled points which do not help to reduce the number of disconnected sub-graphs in the roadmap are ignored. The last ignoring step is eliminated in the simple PRM method (sPRM). As the sampling goes on, the resolution of the roadmap is reduced, and the roadmap gets closer to random start and goal positions. PRM then apply discrete search algorithms to find the shortest path in the roadmap in the query phase. With much fewer nodes used to construct the roadmap, PRM is much more effective than discrete search based methods like A* in exploring high-dimensional spaces. PRMs have been proven to be probabilistically complete [98] in terms of the failure rates decrease to zero when the iterations go on. The sPRM has also been proven to be asymptotic optimal. That is, the cost of the found solution converges to the optimal cost with probability 1 when the number of samples increases to infinity [99]. However, the roadmap constructed by the sPRMs are usually too large to be handled efficiently, especially for large or high dimensional workspaces. [100] has further studied the behavior of PRMs. They have stated that the performance of PRMs relies critically on the visibility properties of the search space. Problems containing narrow passages can be the main bottleneck of the standard PRM. To reduce the computational complexity of PRMs, Bohlin and Kavraki [101] have developed a lazy PRM algorithm. It initially constructs a roadmap without collision information, and delays the checks into the query phase when searching for optimal paths. Colliding nodes found in the focused optimal path are removed from the roadmap and new paths are considered until a collision-free optimal path is found in the roadmap. The lazy PRM has been proven to be still probabilistic complete. Fuzzy PRM [102] is an alternative approach to reduce the number of collision tests. The fuzzy PRM avoids collision detection of edges in the initial construction of the roadmap. Instead, it assigns the estimated probability of collision to the connections, and used these probabilities to guide the selection of paths and the scheduling of collision tests. Later, a bi-directional single-query variation of PRMs has combined the collision detection strategies in [101, 102] to develop a SBL planner (short for Single-query, Bi-directional, Lazy collision-checking planner) [103]. The SBL planner maintains two trees from the initial and goal positions respectively. New points are sampled near sparse areas of

the trees and connected to the expanded node. The other tree then tries to connect with the new vertex if it lies within the neighborhood of the tree. By delaying collision detection, the algorithm has been shown to be very efficient in solving multi-robot problems with many DOFs. Denny et al. [104] have borrowed this lazy collision detection strategy to develop a lazy toggle PRM for single-query purpose. The lazy toggle PRM maintains two graphs: a free graph G_{free} and obstacle graph G_{obs}. Initially, G_{free} performs lazy connects to newly sampled points. When the initial and goal nodes are connected, the feasibility of the paths are checked and invalid nodes and edges are culled from G_{free} and inserted into G_{obs} (witness points are considered for invalid edges). G_{obs} then tries to connect with these new vertices in a non-lazy manner. If two vertices are not able to be connected in G_{obs}, the witness in C_{free} along the path is used to augment G_{free}. The results have shown that the lazy toggle PRM reduces the number of CD calls for several scenarios and maintains the ability to find feasible paths through narrow tunnels from the original toggle PRM. Other trials include the PRM* algorithm [99] developed by Karaman et al. by extending the sPRM method. The PRM* applies adaptive ball radii when choosing nearby vertices to be connected with newly sampled points. This adaptive strategy decreases the ball radius when the number of vertices in the graph have increased, by which the PRM* reduces the number of connection attempts compared to the original sPRM. PRM* reduces the $O(n^2)$ complexity of sPRM to $O(nlog(n))$. The efficiency of PRM* has been further improved in [105] by applying graph spanners to reduce the density of the roadmaps. This algorithm has relaxed the optimality of PRM* and can guarantee convergence to near optimal solutions.

The RRT method (RRT in short) [93] maintains a dense tree expanding from the start position to newly sampled points. RRT expands the position nearest to the sampled collision-free point found in the tree and constructs an edge to connect them using a local planner. In this way, RRT explores C_{free} in a fractal style. Possible connections to the goal position are attempted periodically. The search terminates if the goal is reached. The bi-directional RRT approach (bi-directional RRTs in short) [106] has improved the capability of RRTs to escape from traps. Bi-directional RRTs maintains two trees from the start and end position and extends the trees alternatively.

When one of the trees expands towards a newly sampled point, the other tree tries to approach the same point. As proven by Karaman et al. [98], standard RRTs always converge to sub-optimal solutions. Thus, [107] and [99] introduce a revised version called RRT* which is asymptotic optimal. RRT* first steers the sampled point to the nearest position in the tree and then inserts the steered point as a vertex if it can reach the tree. Afterward, RRT* considers the k-nearest vertices in the tree and selects the one with the minimum cost-to-come to connect with the new vertex. Parents of the remaining vertices are also revised if the new vertex can offer shorter paths towards them. Many improvements of RRTs have been proposed recently. The algorithm in [108] has performed RRT* in an anytime style. The vector field RRT [109] optimizes the upstream criteria of expansion directions. RRT^X [110] refines the path during navigation and repairs its optimal sub-tree when changes happen in the scene.

2.3.3 GA-based path planning

Genetic algorithms are capable of solving complex optimization problems. In the field of robotic path planning, GAs are mainly applied in 2D and 3D navigation guidance in both discrete and continuous spaces.

Many such algorithms use discrete search spaces. The algorithm in [111] uses a GA with variable length chromosomes to search in simple environments represented as 2D grids. It assigns damping coefficients to the grid nodes such that obstacle nodes have positive values and free nodes have negative ones. The fitness function of the GA then optimizes the damping coefficients and distances traveled by the path. Their algorithm has shown effectiveness in both static and dynamic environments. The algorithm in [112] performs binary-coded GA searches in an existing graph with nodes and edges in C_{free}. In this algorithm, the chromosome length is chosen as a function of the number of obstacles in the map based on their observation on the test case. Chung et al. [113] have applied GA in solving TSP in a 3-D grid space. The chromosome length is chosen to be the number of checkpoints. This work has proposed an operator recombining the elite chromosomes in a greedy manner to improve the efficiency of GA for feasible path planning. The work of [114] applies a real-coded GA to search in a coarse-grained graph generated from 2D grids using Dijkstra's algorithm. The algorithm

has achieved similar optimality with better computational performance compared to the original Dijkstra's algorithm. Tuncer et al. [115] have used a decimal-coded GA to search in 2D grids. The GA assigns penalties to infeasible paths and optimizes the total distance of feasible paths. They have also proposed a mutation operator to perform local optimizations. For each gene subject to mutation, the operator selects neighboring free nodes which can reduce the cost of the candidate most to replace the original one.

For continuous search spaces, Tian et al. [116] have applied a binary-coded GA for reverse kinematics path planning to optimize interior points on a 2D polynomial constructed by Hermite cubic interpolations. The problem considered in [116] includes a two-DOF line-shaped robot and several point-shaped obstacles. The results have shown that GA is effective in optimizing joint rotation angles. Shi et al. [117] have also performed GA searches in a continuous 2D space. Their algorithm uses the direction from the initial point to goal point as the z-axis. It distributes via points uniformly on the selected x-direction and regards their y-values as genes. Their fitness function considers path distances, smoothness and security factors. The research in [118] applies GA to optimize the control points of a 2D B-spline using variable length chromosomes. Cakir et al. [119] have utilized a GA with tournament selection mechanism to guide the 2D navigation of Unmanned Aerial Vehicles (UAVs). The algorithm considers UAV navigation as a continuous version of the TSP problem. [120] has also addressed the UAV navigation problem. Their algorithm uses a real-coded multi-population GA to optimize initial paths generated by a greedy heuristic search.

2.3.4 Discussions

Discrete search based path planning algorithms are efficient especially in 2D graphs and grids when dealing with point-shaped or rigid robots. However, when extending to problems with more DOFs like articulated robots, the sizes of the discretized search spaces of these algorithms increase exponentially with the number of DOFs. Therefore, these algorithms can be extremely inefficient for high-DOF problems. Sampling-based path planning algorithms have solved the above-mentioned problem by avoiding explicit representation of the C-space of robots. However, most sampling-based algorithms

only provide feasible solutions. Some of them have achieved asymptotic optimality which ensures to find optimal solutions when the sampling goes on. However, refining the samples leads to increased numbers of vertices in trees or graphs. Thus, due to the memory space limitations, the sampling-based methods are usually used for feasible or near-optimal path planning instead of optimal path planning.

GA-based path planning algorithms can generate higher-quality paths than other types of methods discussed. However, prior explorations on GA-based path planning are mostly restrained within 3 DOFs. Moreover, multi-robot problems have seldom been investigated in previous GA-based algorithms. Part of the reason is due to the exponential size increase of search spaces in higher-DOF problems. Therefore, the GAs may require much more iterations to converge, leading to significant increases in the computation time.

This research explores the superior optimization ability of GA compared to the sampling-based algorithms and, in the meantime, improve the convergence speed and computational performance of GAs for higher DOF-problems.

2.4 Lifting Path Planning

2.4.1 Computer-aided heavy lift planning

Heavy lift planning is an important task in many industries. During occasions like maintenance shutdown and turnaround of industrial sites, hundreds of heavy lifting tasks are required to be conducted within a short time. CALP systems make use of computer simulations and intelligent computations to assist the lift planning process.

Early efforts on CALP focus on developing simulation systems to help in automating common practices in interactive lift planning. Hornaday *et al.* [121] have proposed the their conceptual design of the HeLPS (short for Heavy-Lift Planning System) framework. Lin and Haas [122] have continued the work and have designed a system being able to perform initial setup planning for cranes and performance measurements for user-defined paths. Varghese *et al.* [123] have extended the HeLPS system by monitoring safety factors during interactions. Chadalavada and Varghese [124] have developed their CLPS (short for Critical Lift Planning System) prototype as a plug-in

for the Autodesk Inventor. Their solution enables plant modeling, interactive manipulation, and comprehensive safety monitoring.

Other studies address one or several sub-problems of lifting planning such as crane selection, feasibility checking, and crane layout. Determination of crane locations has been described and attempted as an optimization problem in the work of [125–128]. Olearczyk *et al.* [128] have discussed the crane selection and positioning problem concerning lifting capacities and clearances. Their algorithm solves the crane location determination problem by optimizing weighted distances from crane locations to pick & place locations of the lifting targets constrained by clearances of tail swing, boom and outriggers. Lei *et al.* [126] has also suggested a feasibility checking method for lifting paths by mapping the pick and place areas into the configuration space (C-space) and testing against the obstacle regions (C-obstacle). Another research of Lei *et al.* [127] has discussed the feasibility checking for crawler cranes walking towards distal place locations. Similar to Safouhi's idea in [125], Lei's method dilates the obstacle regions by the size of the lifting target and the tail-swing radius of the crane. Lei's method is able to provide walking paths of crawler cranes as 2D lines with no interference between the dilated obstacles. Sometimes, multiple cranes are required to work together. The algorithm in [129] addresses the multiple tower crane layout problem in construction sites. A hybrid particle bee algorithm is applied to solve the layout problem.

2.4.2 Path planning for single-crane lifting

So far, in all studies and existing systems, automatic lifting path planning has been attempted mostly at the theoretical level with rare implementation reported for practical uses. This is partially caused by that the problem itself is very challenging due to the complexity of plant environments and cranes. The three major concerns of lifting path planning are efficiency, solution quality and success rate. The existing methods use combinations of different search algorithms and collision detection strategies to fulfill the above-mentioned criteria. The first class of methods utilize global optimization search algorithms to achieve high solution qualities. These algorithms are usually combined with pre-computed free spaces. Sivakumar *et al.* [130] have considered the

simplified representations of cranes as planar kinematic chains with two rotational DOFs. SGA is performed on the 2D C-space where pre-computed collision results are imported into the fitness function as violation penalties. Ali *et al.* [1] have applied a two-stage serial GA to search in pre-calculated C-spaces. Both of the GA-based methods are able to achieve highly optimized solutions. However, these methods are impractical due to the computationally intensive nature of GA. Ali's method also suffers from the high computational cost of generating the 3D C-space. As a result, the methods only managed to deal with simple CAD plants.

The second class of methods also rely on pre-computed collision information. Instead of using the global optimization algorithms, this class of methods use fast search algorithms for finding good but not necessarily optimal collision-free lifting paths. Among these algorithms, some also rely on pre-calculated C-spaces to improve the runtime efficiency. The method by Reddy and Varghese [131] represents cranes as linked rigid bodies with three DOFs (swinging, luffing and hoisting). A heuristic depth-first search is performed in the free space. Given a simple CAD plant environment, their planner is able to achieve good solutions as arrays of independent configurations. Their algorithm, however, still requires the substantial time and memory to generate the 3D free space. Chang et al. [2] have used a PRM to generate paths for swinging and luffing in a pre-calculated 2D C-space. The 2D C-space stores the maximum and minimum hoist heights allowed for each pair of swinging and luffing angles. Rule-based hoisting planning are then conducted on the output path by the previous stage. Their algorithm are able to achieve near real-time solution in simple environments. However, the optimality of the paths cannot be guaranteed since the problem has been decomposed. The algorithm in [132] constrains the movement of the lifting target into a single horizontal plane to improve the efficiency of planning. A* search is conducted in the pre-computed 2D ray-arc intersection map. Their algorithm has the problem overly constrained and thus can only produce sub-optimal paths. Other methods do not rely on the pre-computed free space. Instead, collision detection is performed on the fly during the search (which is referred to as "online collision detection" in this thesis). This online collision check strategy is less affected by the number of DOFs and thus can be efficiently extended to high-DOF cranes. Kang et al. [133] have used

bounding spheres to perform online collision detection and CCD for a bi-directional expanding trees in order to solve the lifting path planning problem for tower cranes. Their algorithm involves three sub-phases: path planning for the end effector (lifting target), crane trajectory coordination and trajectory smoothing. This feasible planning algorithm is efficient. But the success rate is restricted due to the overestimated proximity information. The research in [134] has addressed the lifting path planning problem for 7-DOF crawler cranes. The algorithm have utilized a bi-directional RRT and reported reasonable planning time. However, it tends to produce zigzagged paths when most of the DOFs are enabled.

2.4.3 Path planning for dual-crane lifting

Simulation and planning of dual-crane lifting are much more complex than that of single-crane lifting. The automatic path planning of cooperative dual-crane lifting is performed within a known static environment and aims to optimize the motion cost of the dual-crane lifting paths. This problem possesses a highly multi-objective nature due to the multiple constraints involved such as collision avoidance (geometric constraints), crane cooperation (kinematic constraints) and equilibrium of the lifting target (physical constraints).

Different from normal robotic arms or articulated robots, the dual-crane system can be regarded as a cable-driven robot [135] with the lifting target as the end effector. Most discussions on cable-driven robots are on their control and stability analysis [136–138]. It is more challenging performing automatic path planning for these type of robots. In particular, for the dual-crane lifting problem, the cranes, the lifting target, and the ground form a looped kinematic tree. Determining the equilibrium position of the lifting target requires dealing with a highly non-linear system. Earlier efforts to determine the equilibrium states include optimizing potential energies with geometric constrains [139] and assigning ball-in-socket joints to the suspension subsystem [140]. In this research, the equilibrium of the lifting target is modeled by a set of non-linear equations characterizing the geometric and physical constraints of the suspension system.

Prior investigations on automatic path planning of dual-cranes are restrained in simple environments. Early explorations include [141] which applies hill climbing and A* to search in the dual-crane C-space. The algorithm in [1] applies a serial GA to produce highly optimized dual-crane lifting paths. However, their method is prohibited due to the expensive computation cost. The PRM-based method in [2] also deals with dual-crane lifting planning. For each iteration, a configuration is sampled for the major crane and the assistant crane is used to coordinate with it. The major crane and assistant crane are switched after each iteration. In this way the swinging and luffing operations of the cranes are generated. Hoisting planning is then conducted by choosing minimum feasible heights along the dual-crane path. This problem decomposition makes the algorithm efficient but not valid for dual-crane lifting cases requiring tilting of the target.

2.4.4 Discussions

The existing CALP systems have automated some components in lift planning in conventional CAD environments. However, automatic lifting path planning has seldom been touched, especially for cooperative lifting using dual or multiple cranes. Therefore, the previous systems are more suitable for training applications. The targeted system in this research aims to deal with more general industrial environments and provide optimized path suggestions for versatile lifting tasks.

Unlike other high-DOF robotic path planning scenarios that may rely on fast algorithms to achieve good but not necessarily optimal results, crane lifting path planning is eager for global optimization. Thus, in this research, MSPGA [142] is chosen taking advantage of its superior optimization capability inherited from SGA and possibility to be massively parallelized. The MSPGA-based path planners exploit the LGP strategy to deal with the complex search space, which has also been applied in [143] for robotic arms. The planners make use of a hybrid collision detection strategy to balance the pre-processing and online computations. In this strategy, collision detection information for the cranes is pre-computed in the hybrid C-spaces, while online collision checks of the target are performed for each query.

Chapter 3

GPU-based Real-time Collision Detection Engine

3.1 Introduction

Collision detection takes the geometric representations and motion information of scene objects as inputs to calculate Boolean interference and separation distances (or penetration depths) for two or more objects. Path planning and simulations require collision detection to be handled robustly, accurately and efficiently. The proposed 3D collision detection algorithm is designed for both path planning and simulation of crane lifting. It deals with lifting scenes containing one or more dynamic cranes and a static environment to produce Boolean results and proximity warnings. The plants or sites in the scenes can be highly complex. The cranes used can vary in terms of types and structures.

Over the past decade, image-space collision detection has been proven to be capable of handling complex scenes efficiently. Early explorations include the LDIs [47] which represent the intersection volumes as multiple rendered depth images. In their algorithm, a VOI is firstly acquired through performing pairwise AABB intersection tests. Then the two objects are rendered in directions restricted by the VOI. One LDI is thus generated for each object with entry points and leaving points of the object recorded. After the LDIs are read back from the GPU to the CPU side, the actual

This chapter is partially based on the conference paper: P. Cai, C. Indhumathi, Y. Cai, J. Zheng, "A framework of the crane simulator using GPU-based collision detection", Workshop on Serious Game & Simulation, CASA, May 2012, Singapore.

collision detection is conducted by performing intersection tests using the entry points and leaving points. Their work has been extended in [48] by adding face orientation information in order to detect self-collisions. Entries on the z-list with downward face normals are considered as entry points, and those with upward normals are regarded as leaving points. Self-penetrations can also be identified by performing similar entry-leaving pairing test. Research in [49] has further exploited the LDIs by rasterizing the pairs of potentially colliding objects on three orthogonal directions. The LDIs are then used to analyze the repulsion forces on penetrating surfaces. Cai, et al.[144] has proposed an image-space method using multiple projection directions to detect collision between convex objects.

These previous methods require reading back images from the GPU, which is relatively slow due to the asymmetric memory bandwidths in graphic cards. This data read-back can be avoided by performing the whole collision detection process in the GPU and return only the final results to the CPU side. Moreover, for path planning in static environments, instead of performing a rendering process for each object, capturing the information of all static obstacles with a single rasterization can bring obvious performance improvements.

Different from these prior methods which use OpenGL rendering to generate depth images and read them back to the CPU for collision detection, the proposed algorithm in this research avoids this data read-back. It uses CUDA programming to generate Multi-level Depth Maps (MDMs) and performs collision checks in the GPU. Pixels in the MDMs do not necessarily contain equal numbers of data entries. The memory usage is thus reduced by avoiding the allocation of multiple images containing many empty entries in pixels. This chapter first introduces the procedures of the collision detection algorithm including the pre-processing and runtime stages. Then, it illustrates results in the simulation environment and conducts experiments in several test plants to validate the efficiency and scalability of the algorithm. The MDMs are used to represent the complex environments of plants and sites in Chapters 4 and 5 for path planning purpose. In those chapters, the ASVs (Section 4.4.2) and TSSs (Section 5.6) are checked against the MDMs to perform the continuous collision detection of lifting paths. The proposed algorithm also performs as a supporting engine for the simulation and path planning functionalities of the targeted CALP system detailed in Chapter 6.

3.2 Overview of the Engine

The collision detection engine uses a novel MDM representation for the digital plants and sites, and checks the OBBs, ASVs and TSSs of crane components and the lifting target against this representation. This MDM-based method does not require adjacency information or topological constraints of the environment and supports both mesh and point cloud formats.

Figure 3.1 shows the workflow of the proposed collision detection algorithm. The algorithm involves a pre-processing stage and a runtime stage. The first stage generates the MDM for the environment. For simulation purpose, the runtime stage checks the crane component OBBs with the MDM. This process is conducted for each frame in a DCD manner. For path planning purpose, this stage tests the ASVs or TSSs against the MDM. In this case, the algorithm performs CCD for lifting paths.

3.3 Pre-processing Stage

Figure 3.2 illustrates the workflow of the pre-processing stage which converts triangular meshes into MDMs and stores them in the GPU memory for further use in the runtime stage. Inspired by the concept of OpenGL rasterization, a GPU-enabled MDM generator is designed and implemented with CUDA. To enable efficient parallelizations, this generator is further decomposed into two sub-processes: a pre-rasterization process and a rasterization process.

The pre-rasterization process prepares information of candidate triangles for rasterization uses. It maps the AABBs of the triangles on to the x-y plane and calculates the corresponding pixel blocks to be invoked. The edge function parameters of the triangles are also prepared for quick determination of the positional relationship between the pixels and the triangles.

After all the required data are ready in the GPU memory, the rasterization process is then launched to produce the MDM. In this process, the triangular meshes are divided into batches. Each batch contains triangles to be handled parallelly. In order to achieve a compact MDM for memory saving, a counting kernel is firstly launched. The kernel casts vertical rays from the ground to intersect with triangles and records the

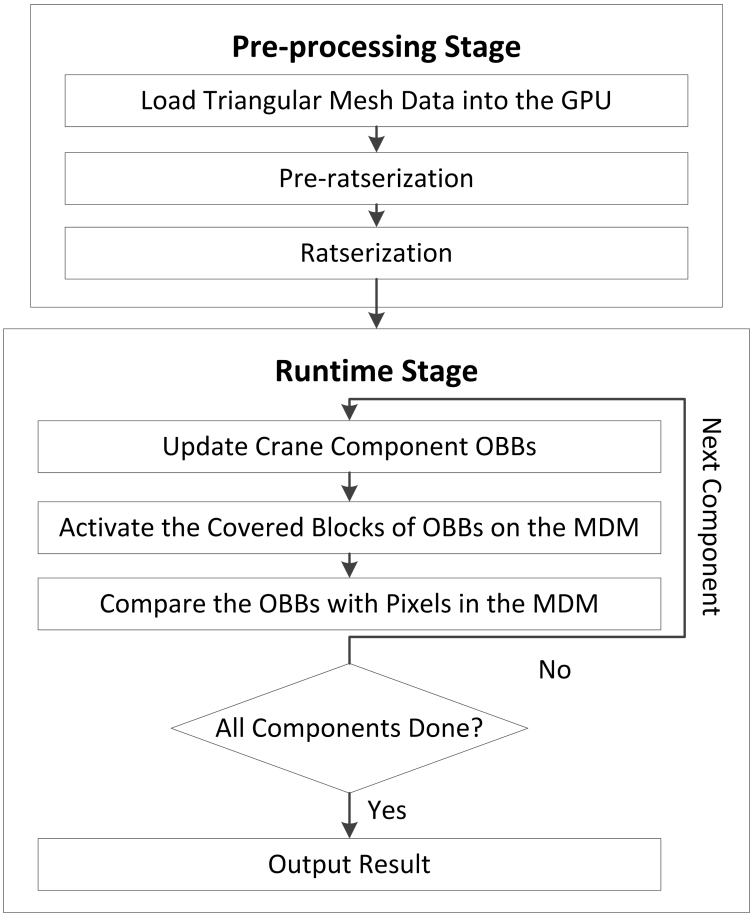


Figure 3.1: Workflow of the collision detection engine.

number of intersections detected in each pixel to create a count map. The MDM is then allocated as a linear array using the total number of intersections acquired from the count map. Afterwards, this generator repeats the ray casting process to fill the MDM entries with intersection heights of the ray and the triangles. This time, each pixel on the x-y plane corresponds to a sub-array of entries from a specific starting position with a pre-counted length in the MDM. The intersection heights with the triangle faces are recorded in the corresponding positions in the MDM. Finally, the generator sorts the arrays in the MDM in an ascending manner, and the process is complete. For some special cases, this generator can be simplified to produce single-level depth maps which only take the highest intersection depth into consideration. In this case, the generator uses atomic functions [52] to prevent conflicts in memory locations. When using the single-level depth maps, the algorithm performs 2.5D collision detection.

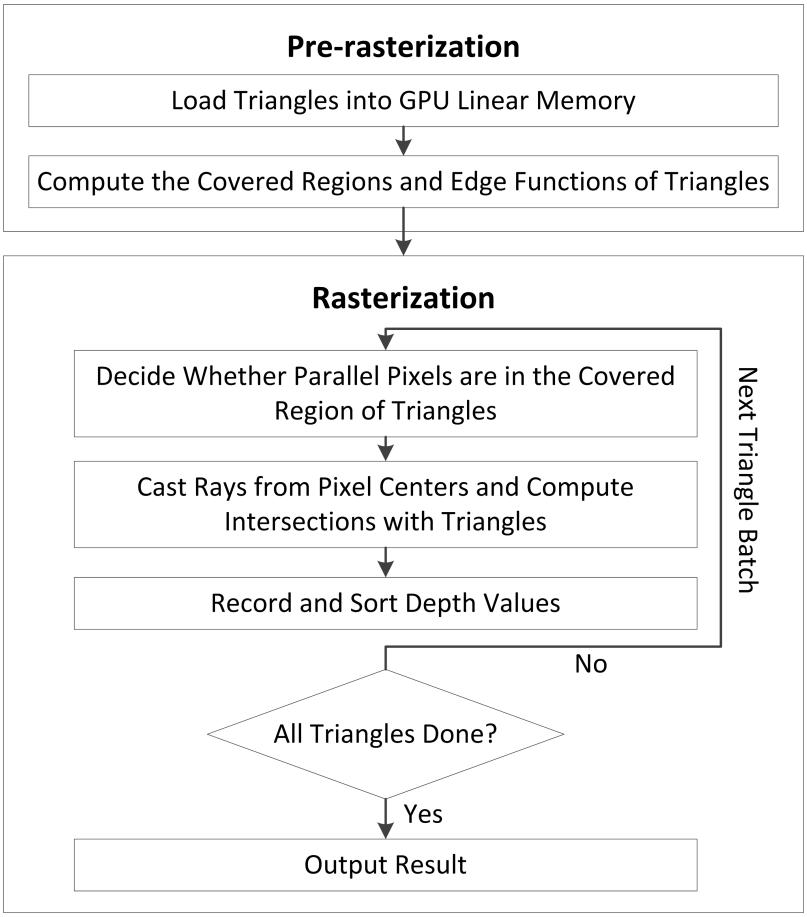


Figure 3.2: Flow chart of the pre-processing stage of the proposed collision detection algorithm.

3.4 Runtime Stage

When the MDM is ready in the GPU memory, the runtime collision check is launched and performed for each simulation time step. The workflow of the runtime stage is shown in Figure 3.3.

In this stage, crane component OBBs are firstly updated according to the configurations of the crane at the time step. Then, pixels in the affected MDM regions of the OBBs are invoked and processed parallelly. For each pixel, a vertical ray is cast to intersect with the OBBs. The intersection intervals of the crane OBBs are then used to compare with the corresponding array of the pixel in the MDM. A collision is reported if an odd number of MDM depth points are found below the crane interval

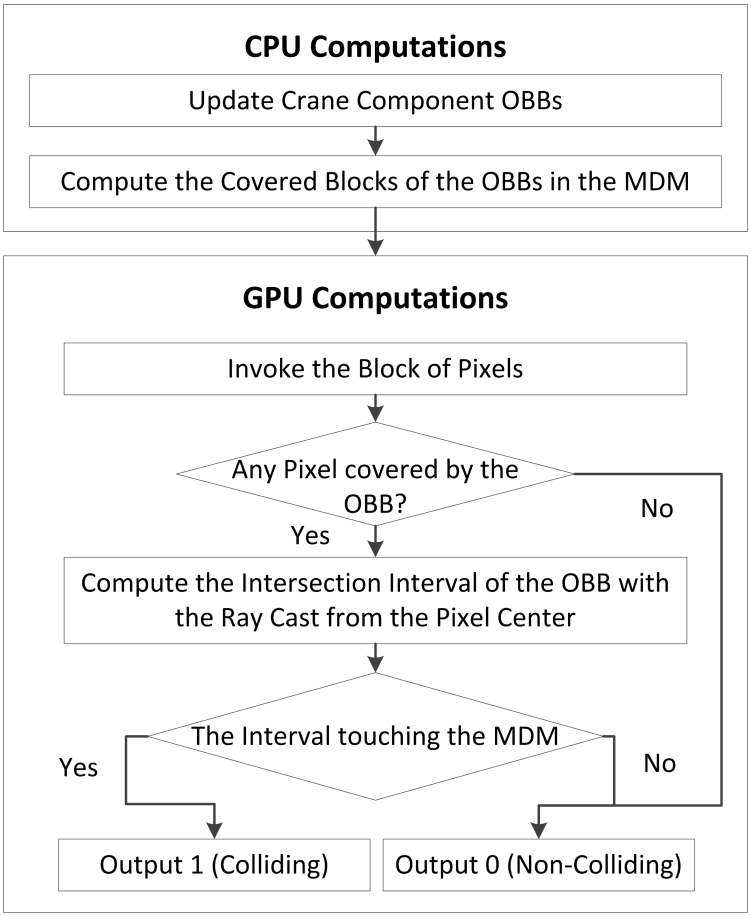


Figure 3.3: Workflow of the runtime stage of the proposed collision detection algorithm.

or if there exist MDM points lying between the crane interval. Otherwise, the pixel is regarded as collision free. Figure 3.4 illustrates this checking process. In case when the MDM is simplified to a single-level depth map for 2.5D collision detection, collisions are reported if any of the MDM points lie above the crane intervals. For path planning purpose, the runtime CCD checks are performed similarly for the ASVs or TSSs of neighboring genes in the GA population (see details in Chapters 4 and 5).

The runtime stage can also produce proximity information using multi-level OBBs. In this case, each crane component is equipped with a tight fitting OBB and two looser OBBs to generate discrete proximity information. In the current system developed, the two looser OBBs are respectively 2 and 6 meters larger than the tight OBB in edge lengths as shown in Figure 3.5.

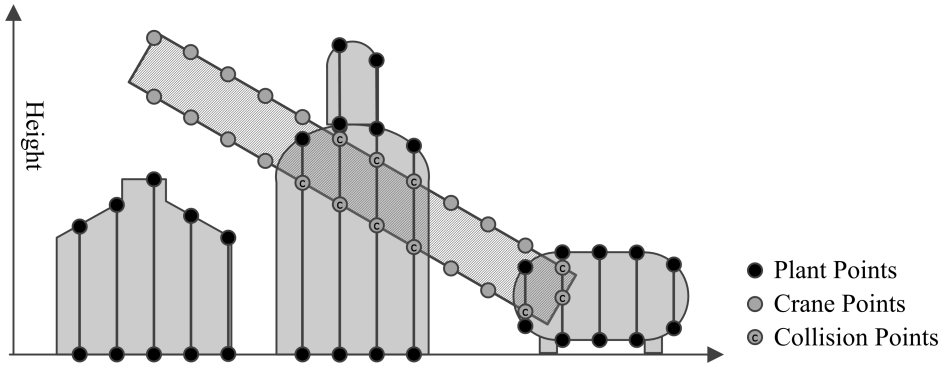
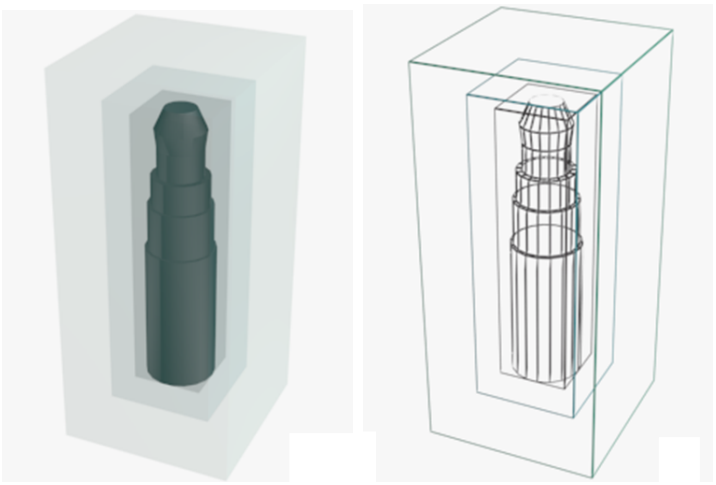


Figure 3.4: Checking the primitives with the multi-level depth map for accurate 3D collision detection



(a) (b)

Figure 3.5: Illustration of multi-level bounding boxes of objects. (a) in solid shape; (b) in wire frame.

3.5 Extension to Point Cloud Environments

Point clouds can digitize 3D environments in high accuracy with detail information. However, they usually contain huge data sets which make it difficult for efficient collision detection. Conventional approaches would require big and complex SP hierarchies to help localize the contacts. The MDM representation proposed in this research can reorganize the point clouds into uniform data structures which is very suitable for collision localization. In this case, points are linked into the corresponding pixels according to its x-y coordinates. Instead of calculating intersection heights, the MDM generator for point clouds directly record the height of the points in the arrays. Compared to

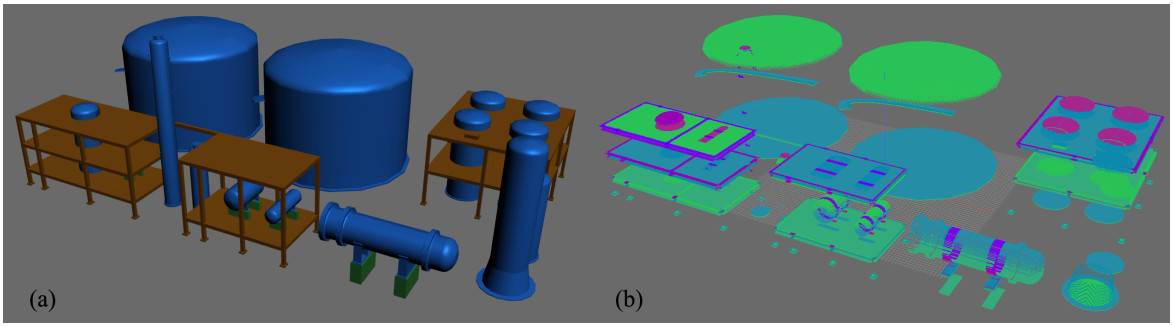


Figure 3.6: Digitization of the plant model using an MDM: (a) Original scene, (b) The MDM generated using the proposed algorithm. Different colors stand for layers in the MDM

the original point cloud data, the MDMs have highly uniform structures and can be efficiently used for collision detection.

3.6 Results and Discussions

This section shows the MDMs generated using the proposed algorithm and demonstrates the collision checking and proximity warning functionalities enabled by the algorithm. Scalability of the proposed collision detection algorithm, in terms of the execution time with environments of different complexities, are also shown in this section.

3.6.1 MDMs of triangular meshes

Figure 3.6 shows the result of the generated MDM compared with the original triangular model. Higher resolution MDMs have more accurate shapes of the environment. On the other hand, lower resolution MDMs lead to less time for both pre-processing and runtime collision checks. Thus, the proposed image-space collision detection algorithm enables users to trade off between the accuracy and the efficiency of collision detection.

3.6.2 Collision check and proximity warning

Figure 3.7 illustrates the collision checking and proximity warning results achieved by the proposed algorithm. If the distance between the lifting target and the plant

structures is within the given proximity thresholds, multiple levels of safety warnings are triggered in the algorithm. When collision happens, a crash sound is played and OBBs of the colliding crane components are shown in red color. In lifting simulation, these warning feature can be used for crane safety training.

3.6.3 Execution time

Four plants with different levels of complexities (Figure 3.8) are used to test the performance of the proposed collision detection algorithm. These plants are selected to ensure that majority of the environment triangles are covered by the crane components and thus take part in the collision detection process. The configuration of the crane is shown in Figure 3.9(a). The MDM of the plant with the highest complexity is shown in Figure 3.9(b). Execution times for both the 2.5D and 3D versions of proposed collision detection algorithm are plotted in Figure 3.10 to show the performance and scalability. When the number of triangles in the plant increases, the execution time of the 2.5D collision detection remains almost the same. This results indicate that, the collision detection components in the path planning algorithms to be introduced in later chapters are independent of the complexities of the environments, making the resulting path planners also unaffected by plant complexities. On the other hand, the execution time of the 3D collision detection shows an almost linear relationship with the number of triangles. Moreover, the time only increases approximately by a factor of 2 when the number of plant triangles has increased for around 5 times. These results show that the proposed collision detection algorithm is highly suitable for complex environments.

3.7 Summary

In this chapter, a GPU-enabled image-space collision detection algorithm for both collision and proximity queries is presented. The algorithm uses OBBs and MDMs to represent the cranes and environments accordingly. For static environments, the collision detection algorithm only requires to perform one rasterization process for all objects. The algorithm has a good scalability regarding the complexity of the

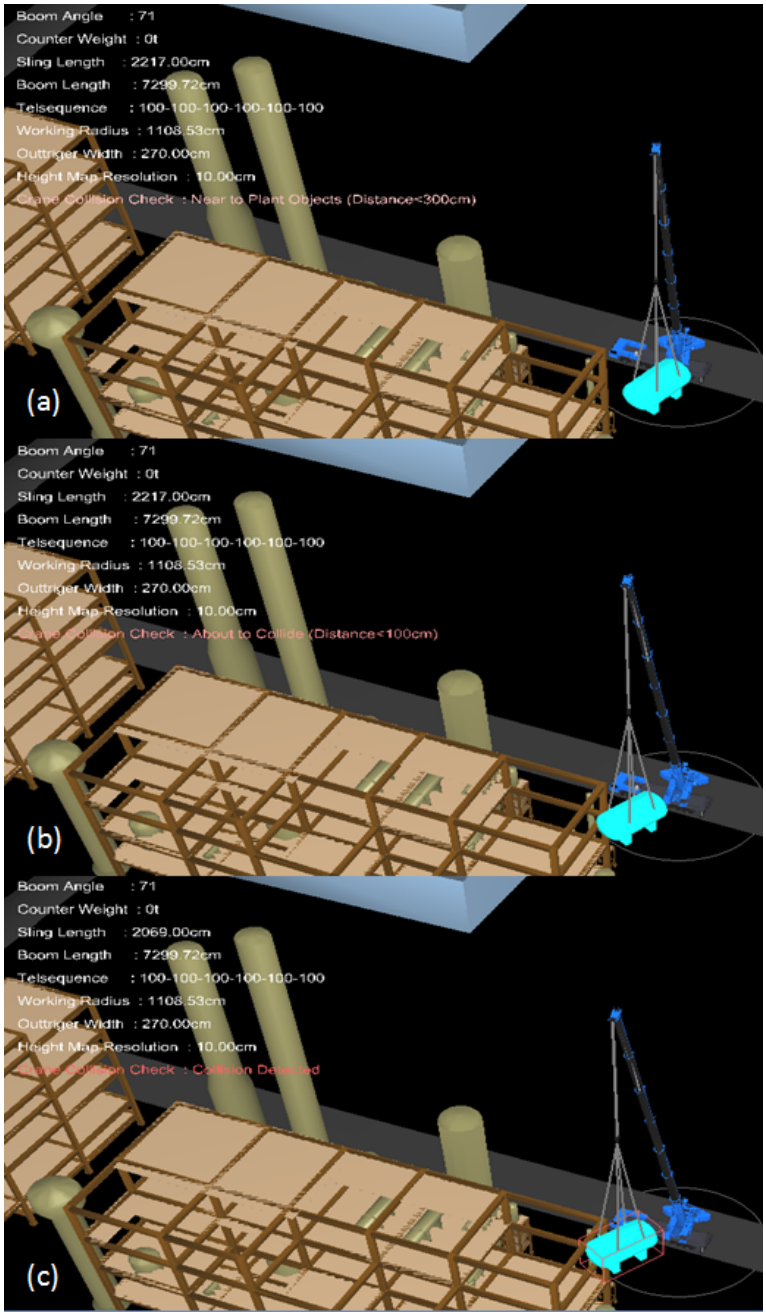
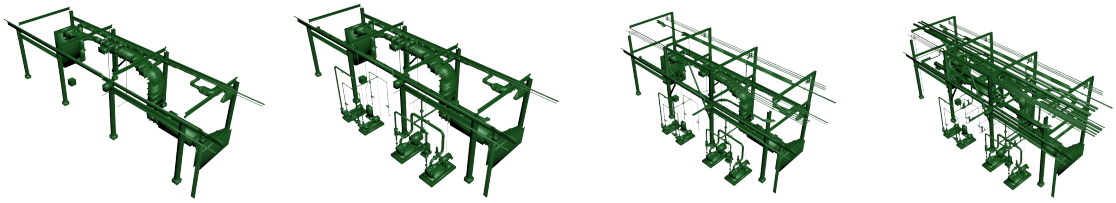


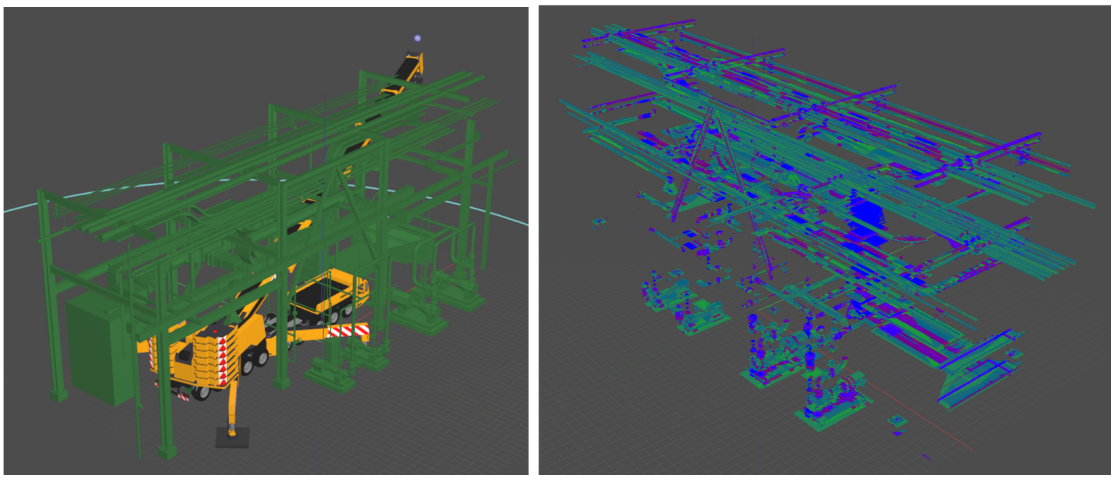
Figure 3.7: Multi-level proximity and collision warning: (a) Distance to plant objects is smaller than 3 meters; (b) Distance to plant objects is smaller than 1 meters; (c) Colliding with plant objects.

environment and is highly suitable for handling complex environments. Particularly, the performance of the 2.5D version of the collision detection algorithm is independent



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

Figure 3.8: Sample plant models for testing the performance of the collision detection engine. (a) test plant 1; (b) test plant 2; (c) test plant 3; (d) test plant 4.



(a) (b)

Figure 3.9: Scalability of the 2.5D and 3D versions of the proposed collision detection algorithm regarding the triangle numbers.

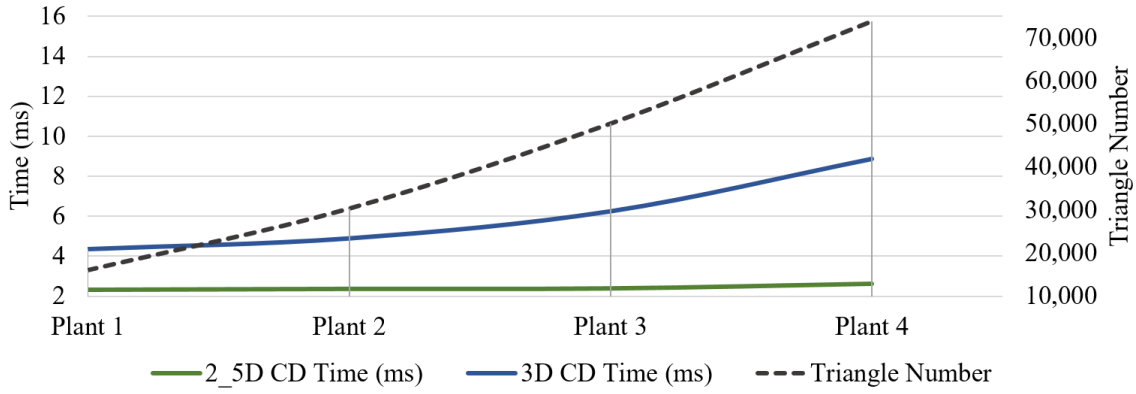


Figure 3.10: Scalability of the 2.5D and 3D versions of the proposed collision detection algorithm regarding the triangle numbers.

of the environment complexities. These features of the proposed algorithm provides a good foundation for the path planning algorithms in Chapters 4 and 5.

Chapter 4

Single-Crane Lifting Path Planning Using GPU-enabled Parallel Genetic Algorithm

4.1 Introduction

This chapter considers the path planning problem for single-crane lifting in complex environments. The cranes involved are terrain cranes. The planning aims to output single-crane lifting paths with optimized motion cost, safety factor and simplicity of operations under constraints of collision avoidance including self collisions. The plant environments considered in this chapter are represented as triangular meshes. The complexity of industrial plants and sites makes it difficult for path planners to achieve high success rates and solution qualities.

Prior researches can be categorized into two classes. The first class of work makes use of pre-computed C-spaces to speed up the runtime performance by sacrificing pre-processing time [1, 2, 131, 132]. This class of methods often utilize global optimum searches in the C-space to acquire high quality paths. Another group of studies rely on an online collision detection strategy which performs collision checks during the searches [39, 133, 134, 145, 146]. Instead of using the global optimization algorithms, these methods use fast search algorithms for finding good but not necessarily optimal

This chapter is based on the journal paper: Panpan Cai, Yiyu Cai, Indhumathi Chandrasekaran, Jianmin Zheng, Parallel genetic algorithm based automatic path planning for crane lifting in complex environments, Automation in Construction, Volume 62, February 2016, Pages 133-147, ISSN 0926-5805, http://dx.doi.org/10.1016/j.autcon.2015.09.007.

collision-free lifting paths. Among the former class, Ali *et al.* [1] have designed a two-stage GA for lifting path planning in simple environments. Their algorithm uses parameter-based reproduction operators for the GA search in pre-computed C-spaces. Their method is able to achieve highly optimized solutions but is computationally forbidding due to the intensive computations required by serial GA. It also suffers from the cost for generating the pre-computed C-space. Another example is the algorithm in [2] that investigates the use of PRM in dealing with crane erection planning. They have proposed a useful idea that, given a 2.5D site with only the maximum height of the plant taken into consideration, a 2D configuration space is enough for the computation. Each position of the 2D configuration space stores the maximum and minimum hoist height of the crane. Hoisting planning is then performed for the 2D path output by the PRM planner. By decomposing the problem, their algorithm is able to achieve near real-time performance in simple environments. Olearczyk *et al.* [132] have tried to conquer the lifting path planning problem by constraining the movement of the lifting target into a single horizontal plane. A* search is conducted in the pre-computed 2D ray-arc intersection map. Their algorithm is fast but relatively impractical because of the planar constraint of the position of the lifting target. The algorithms by [2] and [132] can quickly generate sub-optimal lifting paths. Representatives of the later class include the work of Kang *et al.* [133] who have developed a lifting path planning system using bonding sphere based online collision detection. The bonding spheres are also used in the CCD to detect the interference between the environment and the objects between consecutive time steps. Bi-directional expanding trees are used to search in the Cartesian space for 4-DOF tower cranes. Their algorithm first performs path planning for the end effector (lifting target). A trajectory of the crane is then generated through inverse kinematics. Their method can produce collision-free lifting trajectories in short time. But the success rate is restricted because of the overestimated proximity information and the multiple sub-phases. Lin et.al [134] have introduced their impressive work on the crawler crane lifting path planning problem. The crawler crane is regarded as a robot with seven DOFs. The problem is solved by a bi-directional RRT in a 7D C-space. Their algorithm has reported reasonable planning time. The solution quality is good for low-DOF lifting (≤ 3). However, their algorithm produces more zig-zaged paths for higher-DOF lifting.

In the proposed solution, the terrain cranes are treated as 4-DOF robots. This chapter first provides the mathematical formulation of the path planning problem of single-crane lifting. In order to achieve highly optimized solutions, an MSPGA-based path planner is developed based on this formulation. The planner uses both online collision detection and a pre-computed 2D C-space to trade off the pre-processing and planning query time. The image-space collision detection algorithm introduced in Chapter 3 is combined with ASVs to handle the online collision detection of the crane and the lifting target with the environment. This chapter then introduces a hybrid C-space collision detection strategy to further reduce the planning time. The MSPGA framework and the hybrid C-space strategy also serve as a foundation of the dual-crane path planner in Chapter 5.

4.2 Problem Formulation

This section discusses the mathematical model for the path planning problem of single-crane lifting where the solution space, objective function and constraints are analyzed.

4.2.1 Assumptions

The mathematical model of the problem is established upon the following observations and assumptions:

(i) The terrain crane is not allowed to drive during lifting;

(ii) Booms of the terrain cranes are strictly not allowed to extend or retract during lifting processes.

(iii) The lifting target can be rotated (manually by the rigging man) near the start or end positions.

(iv) It is not permitted to perform the three classes of operations below simultaneously: boom swinging, boom luffing & sling extension, and target rotation.

(v) The speeds of elementary operations are constant (usually very low) and the corresponding energy cost of the crane is proportional to the movement units.

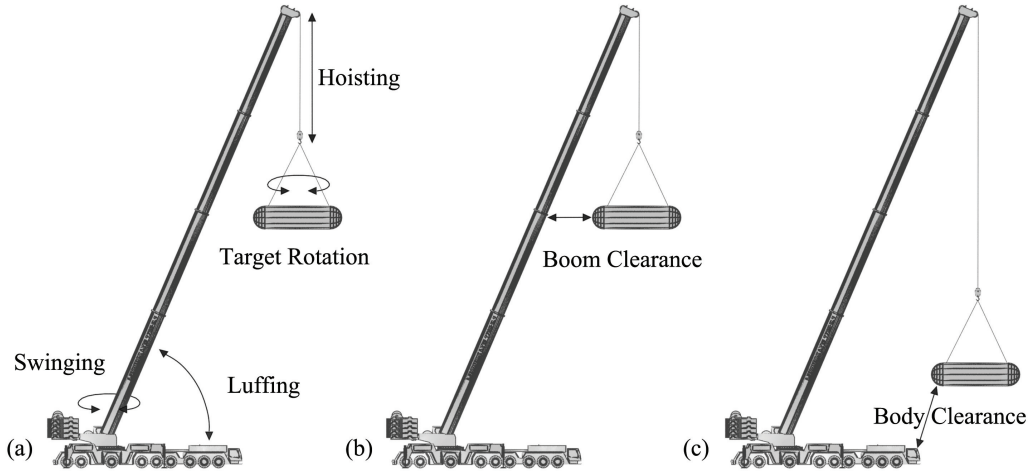


Figure 4.1: The structure of terrain cranes: (a) DOFs of the terrain cranes, (b) boom clearance and (c) body clearance

(vi) Components of the cranes, including the lifting targets, are not supposed to be operated below any plant structures.

4.2.2 Mathematical formulation

According to observations 1, 2 and 3, a terrain crane has four DOFs during lifting operations: boom swinging, boom luffing, hoisting (sling extension & shortening) and target rotation (restrained in start & end positions) (Figure 4.1(a)). Parameters of the DOFs are constrained into limits determined by the internal clearances including boom clearance and body clearance (Figure 4.1(b)). A whole set of parameters specifying the four DOFs form a configuration of the crane. The set of all possible configurations is defined as the configuration space (denoted as C in this paper).

Typically, a crane lifting path is defined as an array of configurations. This configuration array stands for a general poly-line in C that is equivalent to an operation sequence of the crane. However, this definition of lifting path is not going to assure assumption 4 which stated that the operations are decoupled. Therefore, some portions of the poly-line need to be axis-aligned. This constraint requires the GA to handle inter-dependent genes (configurations), arising difficulties in designing parallel crossover and mutation operators.

Table 4.1: Parameters and variables used in solution representation

Symbol	Expression
c_i	The ith configuration in the string
e_j	The jth edge in the string
L_s	Length (number of configurations) of the string
α_{LF}	The luffing angle in degrees (angle between main boom and ground)
α_{SW}	The swinging angle in degrees (rotation angle of main boom along the z axis)
l_{HS}	The hoisting length in centimeters (extension length of the sling)
α_{LR}	The rotation angle of the load along the z axis

To solve this problem, the proposed solution uses independent configurations as genes. Local paths between genes are defined as axis-aligned poly-lines in C. In this way, the genes can be conveniently used by the MSPGA and the local paths between genes are handled in online collision detection and post processing.

In the proposed algorithm, the lifting path is represented as a string $s = \{O, E\}$, where O is the set of nodes (configurations) and E represents the set of edges (internal paths between independent nodes). Thus the variables of the optimization problem can be written as (see in Table 4.1 for explanations of the symbols):

$$s = \{c_i\}_{i=0,1,\ldots,L_s-1} \cup \{e_j\}_{j=0,1,\ldots,L_s-2} \tag{4.1}$$

$$c_i = (\alpha_{LF}, \alpha_{SW}, l_{HS}, \alpha_{LR}) \tag{4.2}$$

The edges e_j are composed by set of key frame configurations determined by its neighboring nodes c_j and c_{j+1} in a pre-defined way which will be discussed in Section 4.3.3. This definition of inputs will be reorganized as chromosomes in the GA search in Section 4.3. The task of the planning algorithm is to find an optimal s^* composed of c_j^* that maximizes the evaluation function.

In order to design an evaluation function for the strings, metric functions are defined in C, so that C becomes a metric space. Those two metrics, d_1 and d_2, can be defined as:

$$d_1(a,b) = \sum\nolimits_{i=0}^{3} r_i|a_i - b_i|; \quad d_2(a,b) = \sum\nolimits_{i=0}^{3} g(a_i - b_i). \tag{4.3}$$

Here a, b denote two configurations in space C and a_i, b_i $(i = 0, \ldots, 3)$ are the unified representationS of the four parameters of a, b (Equation 4.1). Note that a scaling

factor r_i is applied to the absolute difference value for each dimension in d_1. Function g in d_2 is defined as:

$$g(x) = \begin{cases} 1 & \text{if } x \neq 0, x \in R \\ 0 & \text{if } x = 0, x \in R \end{cases} \tag{4.4}$$

Here d_1 measures the total number of movement units along the four dimensions (weighted). d_2 represents the number of non-identical parameters between the two configurations. Now the evaluation function of a string s in the solution space S can be expressed as (see in Table 4.2 for explanations of the symbols):

$$F(s) = \lambda_1 \left(1 + \frac{\lambda_1}{d(s) + \lambda_2(1 + sc(s))}\right) \tag{4.5}$$

where

$$d(s) = \sum_{i=0}^{L_s-2} d_1(c_i, c_{i+1}) \tag{4.6}$$

$$sc(s) = \sum_{i=0}^{L_s-2} d_2(c_i, c_{i+1}) \tag{4.7}$$

$$s \in S, c_i \in C \quad and \quad i = 0, \ldots, L_s - 1 \tag{4.8}$$

Then the maximizing optimization problem for the lifting path planning scenario can be accordingly written as:

Table 4.2: Parameters and variables in the objective function

Symbol	**Expression**
$F(s)$	The evaluation value of string s
$sc(s)$	The operation switching cost in string s
$d(s)$	The distance cost in string s
c_j	The jth configuration in the string
λ_1	The constant scaling factor 1
λ_2	The constant scaling factor 2

$$\max \quad F(s) \tag{4.9}$$

$$s.t. \quad n_{node}(s) = 0,\ s \in S \tag{4.10}$$

$$n_{edge}(s) = 0,\ s \in S \tag{4.11}$$

$$cl(s) = 0,\ s \in S \tag{4.12}$$

$$\underline{B} \leq c_i \leq \overline{B}, \tag{4.13}$$

$$i = 0, 1, \ldots, L_s - 1 \tag{4.14}$$

$$where \quad n_{node}(s) = \sum_{i=0}^{L_s-1} \delta(c_i) \tag{4.15}$$

$$n_{edge}(s) = \sum_{i=0}^{L_s-2} \delta(e_i) \tag{4.16}$$

$$cl(s) = \sum_{i=0}^{L_s-1} \sigma(c_i) \tag{4.17}$$

$$\delta(c_i) \in \{0, 1\}, i = 0, 1, 2 \ldots, L_s - 1 \tag{4.18}$$

$$\delta(e_i) \in \{0, 1\}, i = 0, 1, 2 \ldots, L_s - 2 \tag{4.19}$$

$$\sigma(c_i) \in \{0, 1\}, i = 0, 1, 2 \ldots, L_s - 1 \tag{4.20}$$

Here $\underline{B}$ and $\overline{B}$ stand for the lower and upper bound values for the configurations. $\delta(c_i)$ and $\delta(e_i)$ represent the collision detection results of elements c_i and e_i in string s. $\sigma(c_i)$ stands for internal clearance checking result for configuration c_i. Accordingly, $n_{node}(s)$, $n_{edge}(s)$ and $cl(s)$ represent the collision violation factors of node configurations, edge paths and internal clearance within the crane itself. These values are calculated from the complex geometric information in the Euclidean space.

Feasible solutions of the maximization problem contain no violation for collision and inter-collision. The motion cost $d(s)$ and operation switching cost $sc(s)$ are minimized

through maximizing function $F(s)$, making the optimal solution s^* short in distance and comfortable for human operators. The scale factors enlarge the difference between good solutions and bad solutions, and thus helps the convergence of the GA. As a result, the optimal solution s^* of the maximization problem is a collision-free lifting path which is optimized in energy cost and human operation conformity.

The mathematical and algorithmic details of the computations will be discussed in Section 4.4.

4.2.3 Fitness function

When transferring the optimization problem into the language of GA, hard constraints of the optimization problem (Equation 4.9) are incorporated into the fitness function as penalties. The fitness function for a given chromosome s_i in the population P is defined as (see in Table 4.3 for explanations of the symbols):

$$f(s_i) = \begin{cases} \lambda_1/n_i & \text{if } n_i > 0 \\ \lambda_1(1+\lambda_1/m_i) & \text{if } n_i = 0 \end{cases} \tag{4.21}$$

where

$$n_i = no_i + nf_i + nr_i + nc_i \tag{4.22}$$

$$m_i = d_i + \lambda_2(1 + sc_i) \tag{4.23}$$

$$i = 0, 1, 2, \ldots, L_p - 1 \tag{4.24}$$

When $n_i > 0$, some of the nodes or edges in string s_i are colliding with the environment or the crane itself. In such case, the fitness function focuses on the elimination of collisions. Once $n_i = 0$, it means that s_i becomes a feasible solution. The functionality of GA turns into optimization of the objective function in the feasible space. This fitness function includes the internal clearance as a hard constraint and counts the number of operation switching into the motion cost.

Table 4.3: Parameters and variables used in the fitness function design

Symbol	Expression
$f(s_i)$	The fitness value of string s_i
s_i	The ith string in the population
n_i	The collision violation number
m_i	The motional cost
L_p	The size of population
no_i	The collision violation number of OBBs in string s_i
nf_i	The collision violation number of boom swept volumes in string s_i
nr_i	The collision violation number of load swept volumes in string s_i
nc_i	The collision violation number of internal clearance in string s_i
sc_i	The operation switching count in string s_i

4.3 MSPGA-based Path Planner for the Single-crane Lifting

As a multi-objective non-linear integer optimization model, the lifting path planning problem is challenging for common combinatory optimization methods. GA, as a general optimization algorithm which is mathematically proved to be able to achieve global optimum, is highly suitable for the task. It has good potentials for customization and parallelization. This section presents an in-depth discussion on designing the customized adaptive plan for the path planning problem of single-crane lifting in complex environments. A post processing stage is also introduced to improve the human conformity of the result path. Based on the framework, the collision detection algorithm introduced in Chapter 3 is investigated to deal with discrete and continuous collision detection in complex environments. A hybrid C-space collision strategy is proposed for improving the efficiency of the collision detection module. The collision detection module will be further discussed in Section 4.4. The GPU implementations of the MSPGA-based path planner will be discussed in Chapter 6.

A chromosome (Figure 4.2) in the proposed MSPGA framework is defined as the array of configurations (genes) taking all the node configurations c_i $(i = 0, \ldots, L_s - 1)$ from the solution s in Equation 4.1. The population is thus a set of chromosomes carrying different path candidates evolving in the GA process.

4.3.1 MSPGA framework

The functioning of the MSPGA relies on a complete system with software and hardware components. The software components provide a simulation environment for generating necessary inputs (crane position, boom length, destination configurations, and so on) and displaying outputs (paths, costs, capacities and so on). Figure 4.3 is a brief denotation of the system.

The master processor used for the MSPGA is the CPU. The four functional components, fitness evaluation, selection, crossover and mutation, are handled by the GPU. In each iteration of GA, GPU kernels for the four components are executed in sequence. Among these components, selection and crossover select good candidates from the population to produce offsprings. The mutation process randomly alters genes in offsprings in order to help GA find better configurations in the neighbouring spaces. At the end of each iteration, fitness values of the new population are returned to the CPU to be checked against the termination fitness value. Once the termination fitness is met, the CPU stops the GA process and extracts the optimum chromosome from the GPU memory. If the search exceeds a certain number of iterations, the CPU will stop the search and report failure. As MSPGA preserves the property of probabilistic completeness of SGA, the possibility of failure will decrease to zero when the number of iterations increases.

It is difficult to find a set of termination conditions that guarantee a fully optimized result for a randomized search algorithm like GA. Thus, apart from the basic termination criteria stated above, additional help is sought from the simulation environment. The final solution, or lifting path, with its properties is displayed as animations with real-time monitoring. Based on the graphical results, users can choose whether to conduct further searches.

4.3.2 Adaptive plan

Initialization of the population is the basis of GA. It provides the initial resources and information for the GA to start the search. The proposed initialization uses the strategy shown in Table 4.5. c_0 and c_{L_s-1} in the strings are the start and end configurations of the task. For c_1 and c_{L_s-2}, the hoisting lengths are randomly generated and other

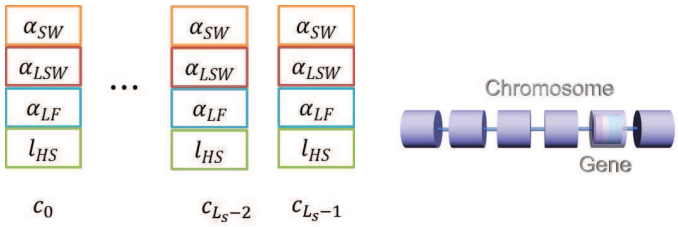


Figure 4.2: Structure of chromosomes in the proposed algorithm

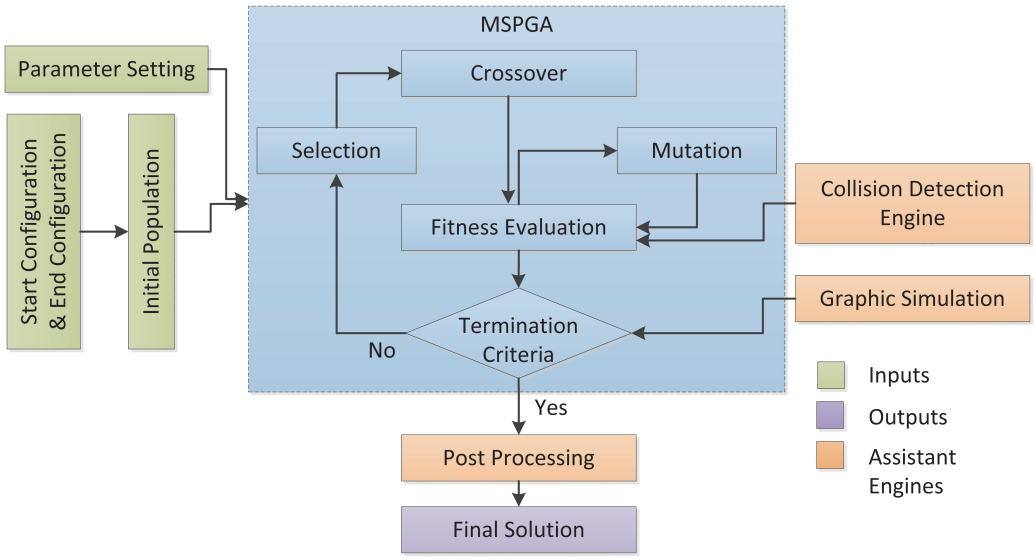


Figure 4.3: Framework of the path planner for single-crane lifting

values are kept as in the start (for c_1) or the end configuration (for c_{L_s-2}). Internal configurations are generated randomly within the bound values.

Selection, crossover and mutation are referred to as the reproductive operators in GA. They are the key components of an adaptive plan which are performed in the evolutionary iterations. By designing proper reproductive operators, the GA can achieve higher convergence speed and, in the meanwhile, produce high quality solutions.

The selection operator reflects the concept of "survival of the fittest" in Darwinian evolution. A good selection operator provides better chance for "fitter" individual to survive and reproduce. In the proposed algorithm, a proportional selection scheme is used together with the elitism strategy. Namely, the "fittest" chromosome in the population always survives and remains in the next generation. For other chromosomes, the chances of producing off-springs are proportional to their fitness values.

Table 4.4: Parameters and variables in adaptive mutation rates

Symbol	**Expression**
$r_m(s)$	The mutation rate of string s
$\overline{r_m}$	The basic mutation rate
$\overline{f}$	The average fitness value in the population
$f(s)$	The fitness value of string s

Crossover happens with a given rate r_c when parent strings are mating. Its mathematical essence is to direct the search to "fitter" areas in the solution space by combining information in existing solutions. In the proposed method, crossover is also responsible for eliminating invalid (colliding) configurations from the population. The proposed approach applies a parameter based crossover strategy as illustrated in Table 4.6. For c_1 and c_{L_s-2}, the off-springs inherit the higher target positions between their parents. In-between configurations inherit valid configurations from the parents in the sense of collision avoidance and internal clearance. The purpose of this strategy is to help GA enter the feasible (collision-free) space through giving higher priority to genes with better potential for collision avoidance.

The mutation operator alters bits (genes) in existing chromosomes with a given rate r_m. Chromosomes with lower fitness values can thus perform as seeds for exploring unknown areas in the solution space. Larger mutation rates can help the GA in finding new possibilities, but at the same time increase the likelihood of damaging existing good chromosomes. On the other hand, lower mutation rates help to preserve known solutions but slow down the convergence.

This analysis leads to adaptive mutation rates. In the proposed approach, the mutation rates for chromosomes are formulated as (see in Table 4.4 for the symbols):

$$r_m(s) = \begin{cases} \overline{r_m} + (\overline{f} - f(s))/\overline{f}, & \text{if } f(s) < \overline{f} \\ \overline{r_m}, & \text{if } f(s) \geq \overline{f} \end{cases} \tag{4.25}$$

The mutation strategy used in the proposed algorithm is shown in Table 4.7. Note that for c_1 and c_{L_s-2} in each chromosome, the mutation only alters the hoisting length (sling length). This design comes from the observation that the first and last step of lifting operations are always hoisting or lowering the target.

Table 4.5: The initialization strategy used in the single-crane path planner

Configuration	Strategy
c_1	Randomly generate sling length; Keep other parameters as the start configuration
$c_2 \sim c_{L_s-3}$	Randomly generate parameters
c_{L_s-2}	Randomly generate sling length; Keep other parameters as the end configuration

Table 4.6: The crossover strategy used in the single-crane path planner

Configuration	Parents	Strategy
c_1	Both valid	Choose shorter sling length
	Both invalid	Random choose
	One valid, one invalid	Choose valid
$c_2 \sim c_{L_s-3}$	Both valid	Random choose
	Both invalid	Random choose
	One valid, one invalid	Choose valid
c_{L_s-2}	Both valid	Choose longer sling length
	Both invalid	Random choose
	One valid, one invalid	Choose valid

Table 4.7: The mutation strategy used in the single-crane path planner

Configuration	Case	Strategy
c_1 & c_{L_s-2}	Valid	Randomly alter sling length in smaller scale
	Invalid	Randomly alter sling length in larger scale
$c_2 \sim c_{L_s-3}$	Valid	Randomly alter all parameters in smaller scale
	Invalid	Randomly alter all parameters in larger scale

4.3.3 Post processing

The task of post processing is to build configurations in the edges from the node configurations returned by the GA search. As mentioned in Section 4.2, the cranes are not allowed to perform the three classes of operations simultaneously. Thus an edge e also contains three segments: boom swinging, luffing & hoisting and load rotation. With the movement units already provided by the node configurations, the key of the edge building strategy is to define the sequence of conducting the three classes of operations.

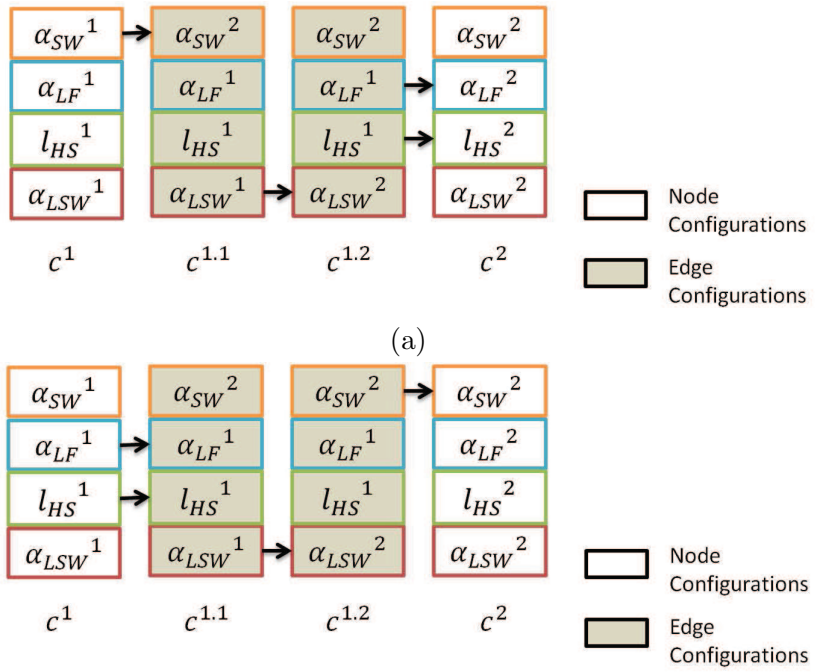


Figure 4.4: The post processing strategy: (a) when the target position in c^1 is higher; (b) when the target position in c^2 is higher.

From observations of lifting operations, when conducting boom swinging, it is safer for the target to be in higher positions than in lower ones. Accordingly, the order of boom swinging and hoisting for a given edge e should be determined by the target height in the two neighbouring node configurations c^1 (left neighbour) and c^2 (right neighbour). For example, if the target is at higher position in the configuration c^1, then boom swinging should be conducted ahead of hoisting. Rotation of the target is located in-between the other two types of operations. Figure 4.4 shows the stated strategy of edge building which identifies the key frame configurations $(c^1, c^{1.1}, c^{1.2}, c^2)$ for edge e.

This strategy is also applied when performing the CCD for edge paths (Section 4.4.2). In this way the solution provided by the GA search will be guaranteed to be valid after the post-processing.

4.4 Collision Avoidance

The proposed algorithm may not rely on pre-computed collision information in the configuration space. The GA, therefore, needs to perform online collision detection when running its iterations. For collision detection of the nodes, the image-space collision detection proposed in Chapter 3 is utilized. This section discusses how this collision detection algorithm is integrated into the context of GA. For CCD of the edges, analytical representations of the swept volumes of the crane are developed and used.

4.4.1 Discrete collision detection

Petrochemical and pharmaceutical plants usually have highly complicated structures. According to assumption 6 in Section 4.2, terrain cranes are not supposed to move or operate below any plant structures. Thus, for a crane, the frontier of contact with the plant structure is determined by the plant structure's highest portions. As such, the plant model is transferred into a histogram shaped structure by computing a single-level depth map of the model. This representation will be used in both the collision detection for nodes and edges in strings. Benefiting from the highly ordered feature of the depth maps, the collision detection can be parallelized in three levels: pixel level, gene level and chromosome level. This three-level parallelization enables the high efficiency of the collision detection components of the planner. Details of the GPU parallelizations will be discussed in Chapter 6.

The procedure of the collision detection of genes in the MSPGA-based path planner is:

(i) Computing an initial OBB hierarchy for the crane model

(ii) Generating the depth map for the plant model

(iii) Updating the crane OBBs for each gene in the population, and

(iv) Performing collision checks between the crane OBBs and the plant depth map for each gene in the population

The OBB hierarchy of a terrain crane contains five OBBs. They are OBBs of body, cockpit, counterweight, boom and load, respectively. Bottom portions of these OBBs are the contact frontiers to the plant depth map. As the hook and slings are always above the load for terrain cranes, they are not going to affect the contact frontier. During the GA search, each gene possesses one such set of OBBs. In each iteration, the set of OBBs are updated in the GPU parallelly for the whole population. In the stage of fitness evaluation, another GPU kernel is launched to compute the contact frontiers for the OBBs in parallel and compares the OBB frontiers with points in the depth map. The Boolean results returned by the kernel are used to calculate *no* values in the fitness function (see Equation 4.21).

4.4.2 Continuous collision detection

In order to enable sparsely sampled paths and reduce the number of steps in the lifting path, collision detections for the internal trajectories between consecutive configurations become crucial. This goal is achieved by applying the CCD technology using analytical estimations of the swept volumes between consecutive configurations.

During the movements along the internal paths between neighboring genes, the booms and the lifting target have the highest possibility to collide with the plant structure. When the crane moves from one configuration to another, the spaces that the booms and the load swept through are called swept volumes. Bottom faces of these swept volumes are denoted as swept frontiers that are used to perform contact check with the plant depth map. By applying the pre-defined movement strategy (Section 4.3.3), a unique swept frontier can always be obtained for each pair of neighboring genes. Figure 4.5 shows the 2D illustrations of the swept frontiers. In the illustrated case, the crane first performs swinging in the counter-clockwise direction, then rotates the target and lowers down the boom to reach the destination working radius. Analytically, the swept frontiers for this case are represented as (see in Table 4.8 for

explanations of the symbols):

$$A_{TG} = A_{TG\_SW} \cup A_{TG\_LF} \cup A_{TG\_LR} \tag{4.26}$$

$$A_{TG\_SW} = \big(\{d_{xy}(p, \overline{p_0p_1}) \leq d_{11}\} \cup \{d_{xy}(p, \overline{p_0p_2}) \leq d_{11}\} \cup \{\alpha^1_{SW} \leq \alpha(\overline{p_0p}) \leq \alpha^2_{SW}\}\big) \cap \{R_1 - d_{12} \leq p_{xy}(p - p_0) \leq R_1 + d_{12}\} \cap \{z = h_{TG}\} \tag{4.27}$$

$$A_{TG\_LF} = \{d_{xy}(p, \overline{p_0p_2}) \leq d_{21}\} \cap \{R_2 - d_{22} \leq p_{xy}(p - p_0) \leq R_2 + d_{22}\} \cap \left\{z = z_c + \sqrt{L^2_{BM} - (x - x_0)^2 - (y - y_0)^2}\right\} \tag{4.28}$$

$$A_{TG\_LR} = \{p_{xy}(p - p_R) \leq R_R\} \cap \{z = h_{TG}\} \tag{4.29}$$

$$A_{BM} = \{\|p - p_0\| \leq L_{BM}\} \cap \{\alpha^1_{SW} \leq \alpha(\overline{p_0p}) \leq \alpha^2_{SW}\} \cap \big\{z = z_0 + tan(\alpha_{LF})\left((x - x_0)^2 + (y - y_0)^2\right)\big\} \tag{4.30}$$

Here function d_{xy} gets the distance between a point (the first parameter) and a line (the second parameter) on the x-y plane. Function p_{xy} represents the projection length of a vector onto the x-y plane. Function α calculates the planar angle between a given vector and the x axis. In each loop of fitness evaluation, these swept frontiers are compared with points in the depth map of the plant model. The Boolean CCD results are used to calculate nf and nr values in the fitness function (Equation 4.21).

4.4.3 Self-collision clearance

Apart from collision between the crane and environment objects, internal collisions, especially interferences between the lifting target and crane components, are also crucial.

Two types of possible internal contacts are considered (Figure 4.6) (see Table 4.9 for the explanations of symbols). The first type is the clearance between the target and boom segments. This type of clearance check becomes more important when the sling length gets shorter. The mathematical representation would be:

$$cot(\alpha_{LF})(l_{HS} + l_{RG}) > r_{TG} \tag{4.31}$$

The second type of internal clearance is the clearance between the target and the crane body. This type of check is significant when the target is moving in lower positions. The mathematical representation can be written as:

$$h_{TG} > \delta_{BD},\ if\ r_w \leq \gamma_{BD} + r_{TG} \tag{4.32}$$

Table 4.8: Parameters and variables in the swept frontiers

Symbol	Expression
A_{TG}	The swept frontier for the lifting target
A_{TG_SW}	The swept frontier for the lifting target during swinging
A_{TG_LF}	The swept frontier for the lifting target during luffing
A_{TG_LR}	The swept frontier for the lifting target during target rotation
A_{BM}	The swept frontier of booms
p	A point on the swept frontier
p_0	Center of rotation of the main boom
x_0	X coordinate value of p_0
y_0	Y coordinate value of p_0
z_0	Z coordinate value of p_0
z_c	Height of the virtual center of rotation of the lifting target during luffing
p_1	The location of load center at the first swinging angle
p_2	The location of load center at the second swinging angle
p_R	The location at which load rotation happens
α_{LF}	The constant luffing angle during swinging in the internal path
α_{SW}^1	The swinging angle in gene 1
α_{SW}^2	The swinging angle in gene 2
d_{11}	Width of the lifting target at the tangential direction in gene 1
d_{12}	Width of the lifting target at radial direction in gene 1
d_{21}	Width of the lifting target at the tangential direction in gene 2
d_{22}	Width of the lifting target at radial direction in gene 2
R_1	The working radius of the crane at gene1
R_2	The working radius of the crane at gene2
R_R	The working radius at which load rotation happens
h_{TG}	The height of the target during swinging
L_{BM}	The total length of the boom

The clearance test results are used to calculate nc values in the fitness function (see Equation 4.21)

4.4.4 Hybrid C-space strategy

The analytical swept frontier based CCD proposed in Section 4.4.2 has good potential scalability in terms of the DOFs of the cranes. However, it will also result in longer planning time compared to the C-space based approaches. For cranes with four DOFs,

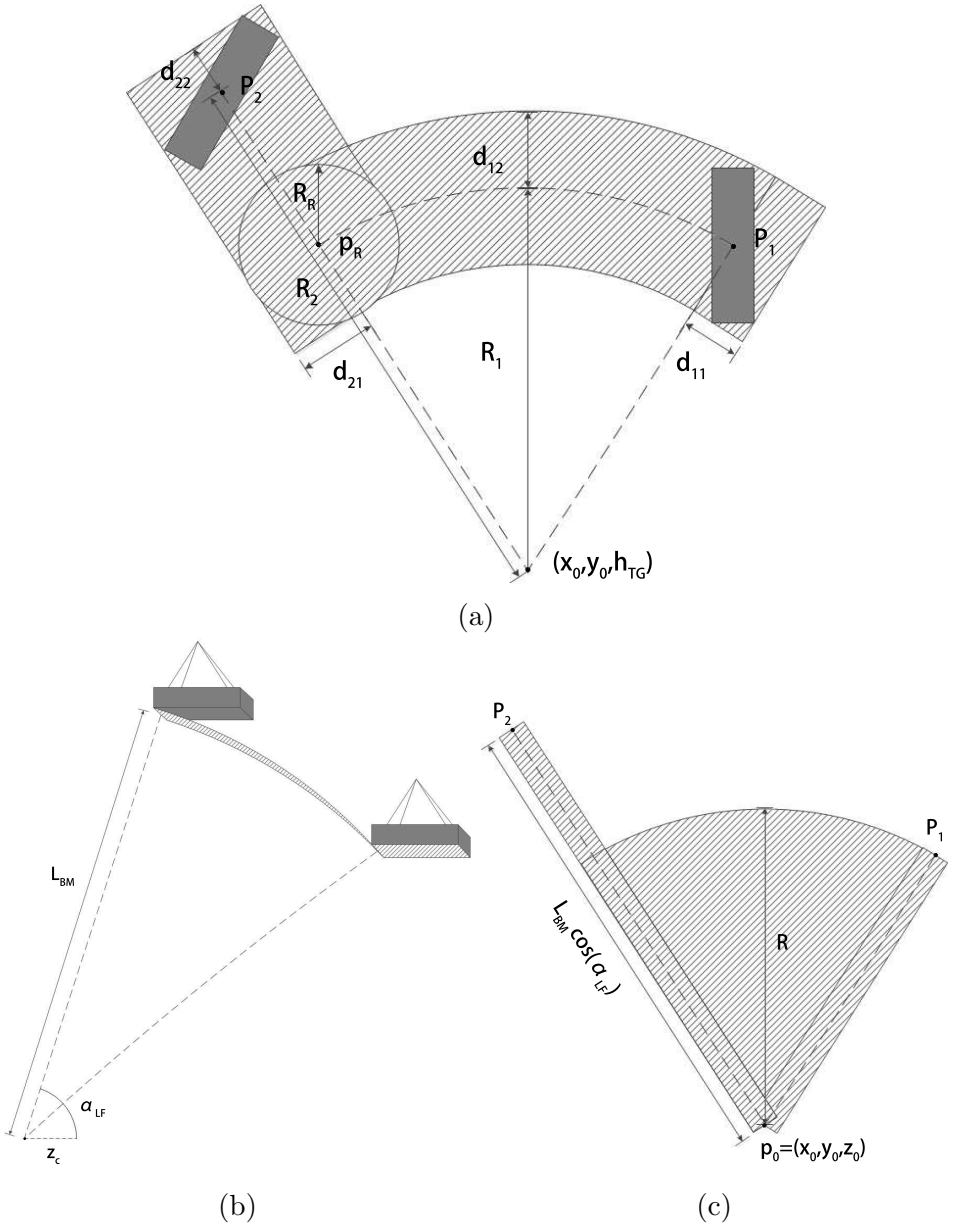


Figure 4.5: Demonstration of the swept frontier of the swinging and luffing operations: (a) target swept frontier from top view; (b) target swept frontier of luffing from side view; (c) boom swept frontier from top view.

the 2D C-space with minimum-maximum hoisting values proposed by [2] can be a good solution for reducing the planning time. But this approach ignored the fourth DOF of the crane (load rotation). Thus it results in over-estimated proximity information in the 2D C-space. For lifting cases where the crane has to rotate the load into a certain

Table 4.9: Parameters and variables in the internal clearance inequalities

Symbol	Expression
α_{LF}	The main boom angle (luffing angle)
l_{HS}	The length of the sling (hoisting length)
l_{RG}	The distance between bottom tip of the hook and center of the target
r_{TG}	The radius of the bounding sphere of the target
h_{TG}	The height of the bottom face of the target
$\delta_B D$	The height of the crane body
r_w	The working radius of the crane
γ_{BD}	The distance between head of the crane body and the rotational axis of booms

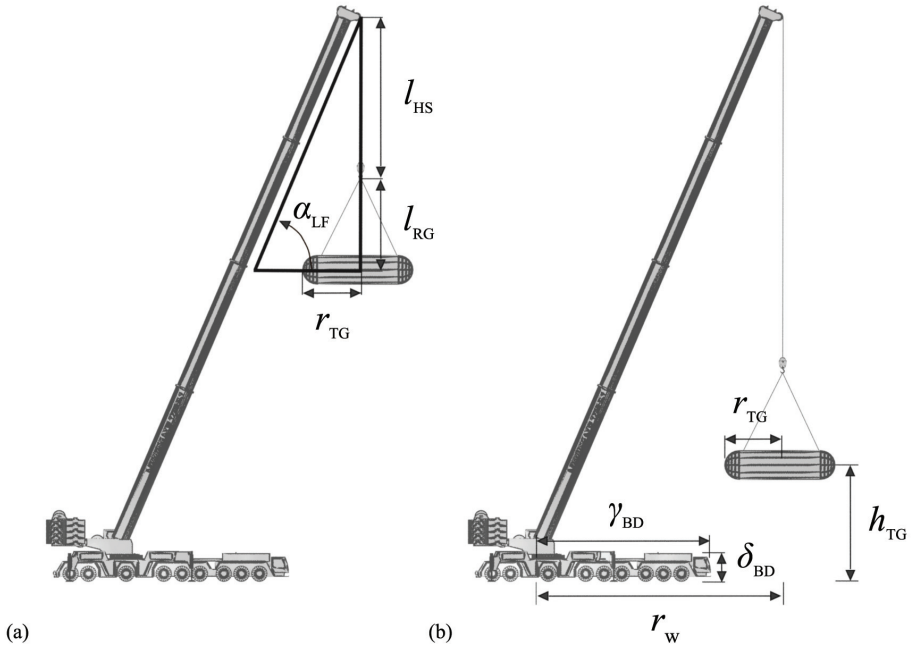


Figure 4.6: Demonstration of variables used in the computation of internal clearance for terrain cranes

angle to avoid the obstacles, this approach will have low success rates.

In this section, a hybrid C-space strategy is proposed to trade off the pre-processing time and the planning time. In this strategy, collision information for the first two DOFs of the crane (swinging and luffing) are stored in a 2D C-space while the collision check (DCD & CCD) for the last two DOFs (hoisting and load rotation) are conducted

during the MSPGA search. Internal clearance check is also conducted during the search. Given the simplicity of the 2D C-space, the hybrid strategy would be able to acquire the C-space in a short time and in the meantime reduce the planning time.

4.5 Results and Analysis

4.5.1 Comparison on the fitness function

To evaluate the fitness function in the proposed method, it is compared with the fitness function used in Ali's work [1]. Their algorithm is initially targeting at the dual-crane erection problem, but the fitness function can be easily adapted to the single-crane lifting problem. By eliminating the coordination violation coefficient in their fitness function (which is not necessary in the single-crane case), a single-crane version of Ali's fitness function is obtained:

$$F(s) = \left(\frac{\lambda}{d(s)(1+C)}\right) \tag{4.33}$$

$$d(s) = \sum_{i=1}^{L_s-1}\left(\sum_{j=1}^{4}(L_{i,j}-L_{i+1,j})^2\right)^{\frac{1}{2}} \tag{4.34}$$

$$C = \frac{1}{L_s}\sum_{i=1}^{L_s} C_i \tag{4.35}$$

Parameters used in Equation 4.33 are identical as in the proposed fitness function. Experiment 4.1.1 is performed in one of the test plants in Ali's paper and compared the success rates and solution qualities. All the runs are performed in the same GA procedure as stated in the Section 4.3 with fitness function differed. The inputs of Experiment 4.1.1 are shown in Table 4.10.

Figure 4.7 shows the result paths using the two fitness functions. The elementary operations are displayed in different colors. The red color dotted line denotes the trajectory performed by the target during the swinging operations. The green and blue colored lines stand for the load center trajectory for the luffing and hoisting movements accordingly. The yellow fan represents the rotation of the load. In the path generated with Ali's fitness function, the crane undergoes 67 degrees of swinging, 10 degrees of luffing, 23.74 meters of hoisting and 67 degrees of load rotation above the

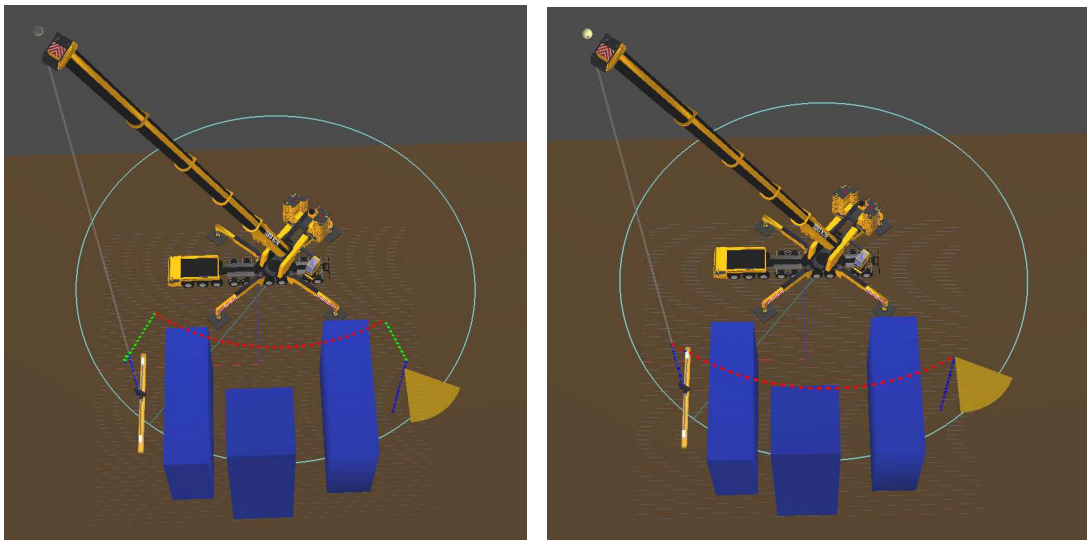
Table 4.10: Inputs into Experiment 4.1.1

Input	**Value**
Crane model Terex	AC700
Size of population	100
Length of string	6
Crossover rate (r_c)	0.15
Mutation rate (r_m)	0.75
Scale of mutation 1	0.016
Scale of mutation 2	0.16
Boom length (m)	62.4
Start configuration	$(67, 119, 4972, 119)$
End configuration	$(67, 52, 4972, 52)$
Termination iteration	400

start position. The path consists of 9 configurations. In the path generated with the proposed fitness function, the terrain crane need to conduct 67 degrees of swinging, 0 degrees of luffing, 26.52 meters of hoisting and 67 degrees of load rotation. The human operators have 2 less steps to conduct and there is no luffing operation involved. The results shows that the proposed fitness function can produce better solution quality.

The convergence trends using the two fitness functions are shown in Figure 4.8 with the fitness value of the best candidate and the average fitness value of the population during the iterations plotted. The convergence of the GA search with Ali's fitness function shows a highly unstable trend. This instability results in a very low success rate (8%). The convergence with the proposed fitness function shows a much more stable pattern. The search achieves convergence within 250 iterations in this test. The success rate reaches 100% using the proposed fitness function.

Similar comparisons (Experiment 4.1.2) are conducted in three additional plants as shown in Figure 4.9. As indicated in Table 4.11, the success rates using Ali's fitness function of lifting cases in plants 1 and 3 are very low (8% or lower) for 50 trials. The success rate of the lifting case in plant 2 using Ali's fitness function is much higher (88%) than in plants 1 and 3. A possible reason is that, the place position of the lifting target in plant 2 is higher than the obstacles, making it easy to achieve collision avoidance. For the lifting case in plant 3 where the obstacles are much higher than



(a) Result path generated using Ali's fitness function: extra luffing (as shown in green colored lines) required.

(b) Result path generated using the proposed fitness function: no luffing required.

Figure 4.7: Result path generated using different fitness functions in Experiment 4.1.1

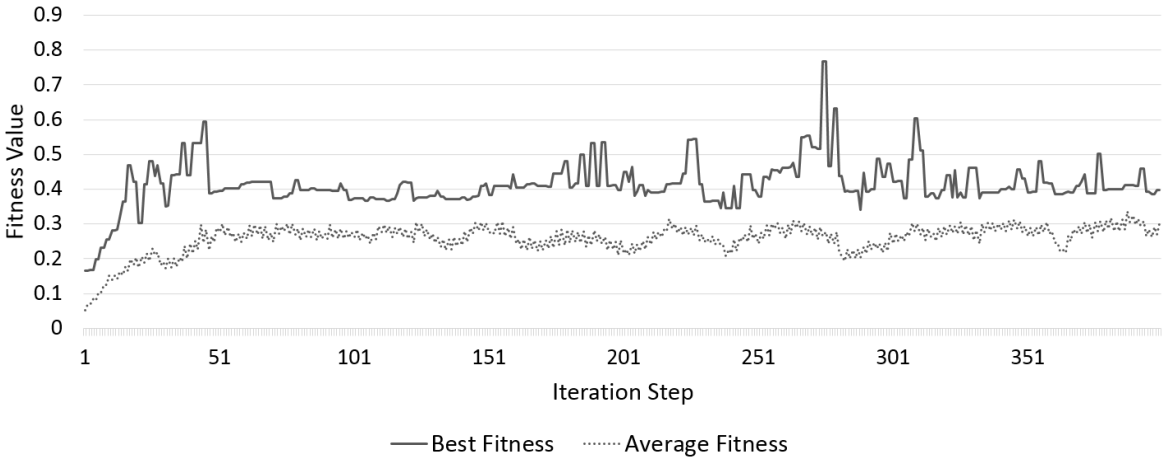
Table 4.11: Comparison of the success rates in the three plants using the fitness function in [1] and the proposed fitness function in Experiment 4.1.2

	Success rate	
	Ali's fitness function	The proposed fitness function
Plant 1	8%	100%
Plant 2	88%	100%
Plant 3	0%	96%

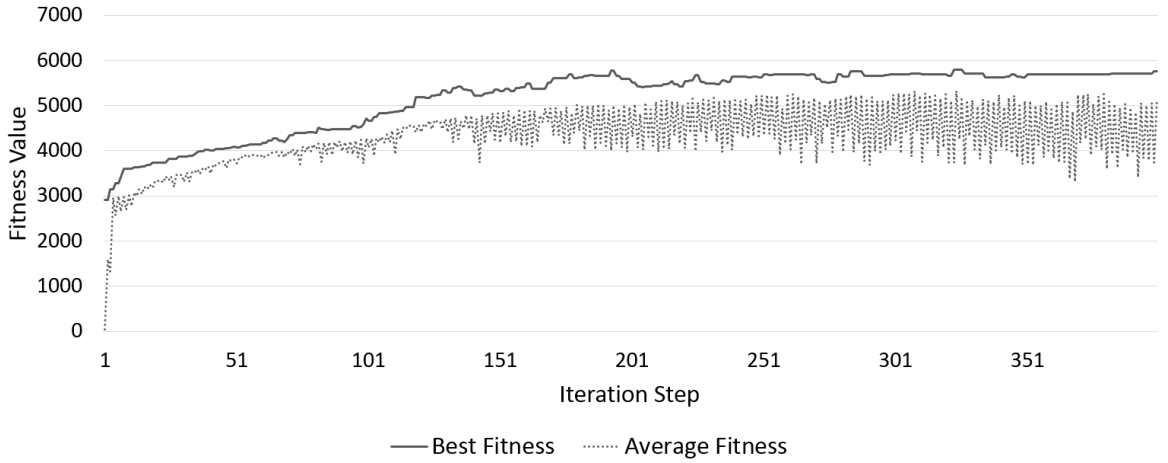
the target pick and place positions, it is more challenging to find valid paths. The proposed fitness function is able to achieve nearly 100% success rate for all the cases.

Table 4.12 shows the average movement units and number of operation steps of the result paths using Ali's fitness function and the proposed fitness function for 50 trials. Although the success rates of the lifting case in plant 2 are both high using the two fitness functions, the solution qualities are quite different. The path generated with the proposed fitness function performs around 16 less degrees of luffing while the hoisting height remains similar. For plant 1 and plant 2, the proposed fitness function can produce paths with 2∼4 less operation steps.

The comparisons indicate that the fitness function use by Ali *et al.* may not

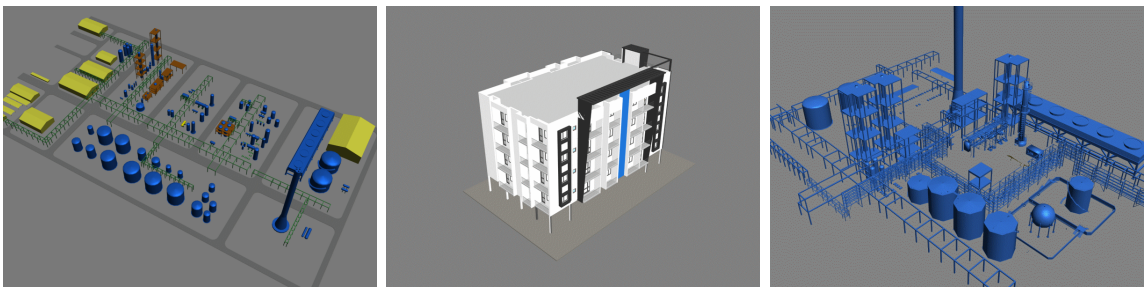


(a) Fitness convergence trend using Ali's fitness function



(b) Fitness convergence trend using the proposed fitness function

Figure 4.8: Fitness convergence trend using different fitness functions in Experiment 4.1.1



(a) Plant 1 (b) Plant 2 (c) Plant 3

Figure 4.9: Additional plants used in Experiment 4.1.2.

Table 4.12: Solution qualities in the three plants using the fitness function in [1] and the proposed fitness function in Experiment 4.1.2

	Solution quality				
			Plant 1	Plant 2	Plant 3
Alis fitness function	Motion units	Swinging (degree)	93	131	-
		Luffing (degree)	8.22	17.19	-
		Hoisting (m)	14.34	20.92	-
		Target Rotation (degree)	93	131	-
	Operation steps		10.67	12.14	-
The proposed fitness function	Motion units	Swinging (degree)	93	131	83
		Luffing (degree)	1.2	1.28	28.4
		Hoisting (m)	17.29	21.45	50.29
		Target Rotation (degree)	93	131	8
	Operation steps		6.86	10.08	9.68

be suitable for the MSPGA framework for single-crane lifting path planning. The proposed fitness function can achieve significant improvement on the success rate of the GA search and in the meantime produce better solution qualities (fewer motion units and operation steps). The improvements are due to three major reasons:

(i) The separation of collision avoidance and optimization of path quality. The fitness value of collision-free paths are much higher than invalid paths. This enables the valid paths fast conquer the population;

(ii) Taking into consideration of the human operational conformities and easiness (fewer operation steps involved);

(iii) Scaling of the fitness function. Big scaling factors in the fitness function increase the difference between valid paths, which increase the selection pressure in the population and push the GA towards convergence.

4.5.2 Validation of the hybrid C-space strategy

To elaborate the benefits of the hybrid C-space strategy (hybrid strategy in short for this section), the proposed algorithm is applied in a complex industrial site to

Table 4.13: Inputs into Experiment 4.2

Input	**Value**
Crane model Terex	AC700
Size of population	100
Length of string	6
Crossover rate (r_c)	0.15
Mutation rate (r_m)	0.75
Scale of mutation 1	0.016
Scale of mutation 2	0.16
Weight of boon swinging	1.0
Weight of boom luffing	1.5
Weight of sling extension	0.06
Weight of target rotation	1.0
Boom length (m)	62.4
Start configuration	$(55, 349, 4212, 349)$
End configuration	$(52, 72, 4001, 341)$
Termination iteration	200

compare the hybrid strategy with the other strategies: C-space and online. The C-space strategy uses the 2D C-space described in [2] and the online strategy uses the analytical swept frontiers to perform online CCD during the GA search as described in Section 4.4.2. The hybrid strategy uses a 2D C-space to store precomputed collision information for swinging and luffing, and applies the analytical swept frontiers for load movements (which reflect the other two DOFs). Experiment 4.2 is conducted in a complex plant containing 274,108 vertices and 376,205 triangle faces. With the same set of experiment setting (Table 4.13), the average computation time, success rates and solution qualities for 100 trial runs are indicated in Table 4.14. The hybrid strategy spends only half the planning time of the online strategy and obtains similar solution qualities. The preprocessing time is also reduced compared with the C-space strategy. Figure 4.10 shows sample result paths generated with the three strategies. While both the online and hybrid strategies manage to rotate the load in order to pass through the plant structure, the C-space strategy needs to pass through the other direction which requires much more movements.

Table 4.14: Results of Experiment 4.2

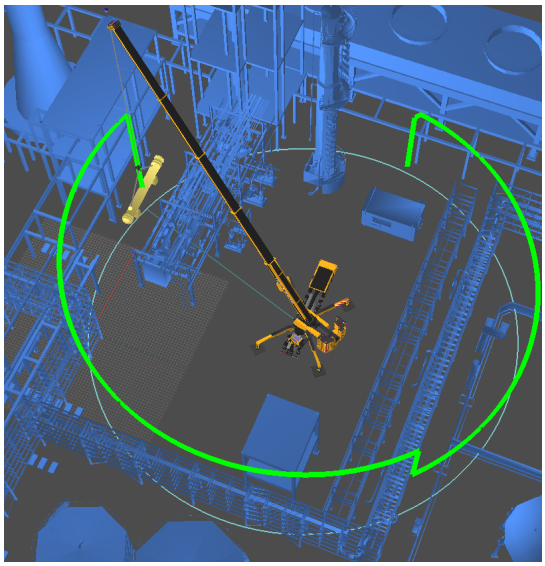
Strategy	**Preprocessing time (ms)**	**Planning time (ms)**	**Success rate**	**Average success fitness**
C-space	10535	789.33	66%	3611.06
Online	0	3998.75	93%	3943.66
Hybrid	1034	2133.79	92%	3965.83

4.5.3 Discussion on parameter design

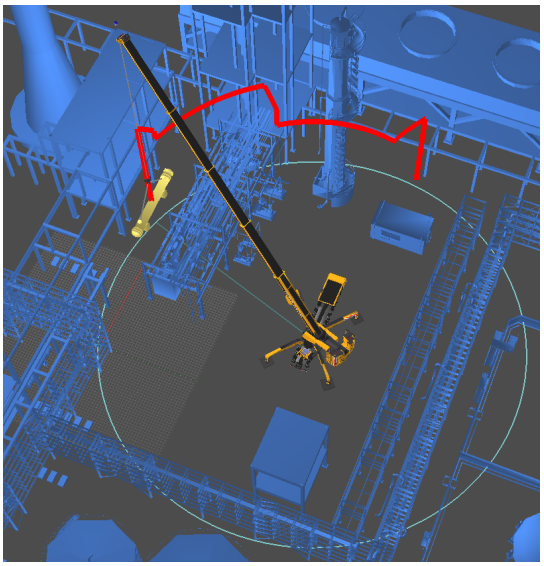
In fact, the selection of parameters, especially the reproduction rates, affect significantly the performance of the algorithm. Experiment 4.3.1 conducts a statistical study for the reproductive rates (r_c & r_m). Using the plant model as shown in Figure 4.11(a), 50 samples of execution are conducted for each combination of reproductive rates. The reproduction rates are spread from 0.05 to 0.75. The results, in terms of fitness value of the final solution and the success rates (probability of obtaining collision-free paths), are illustrated in Tables 4.15-4.16 and Figure 4.12. In this particular plant model, the best solution is achieved with $r_c = 0.25$ and $r_m = 0.55$. Meanwhile, the best ability of collision avoidance is obtained with the crossover rate and mutation rate set as 0.25 and 0.75. It comes to a conclusion that, for this particular plant, the best setting of reproduction rates is: $r_c = 0.25$, $r_m = 0.55 \sim 0.75$.

But the result above is not necessarily suitable for other plant environments. Thus, another experiment (Experiment 4.3.2) is conducted. The height of the major obstacles is reduced in the previous plant so that the crane does not need to raise its booms to avoid them (Figure 4.11(b)). The experiment is conducted with the same set of parameters and the results are listed in Tables 4.17-4.18 and Figure 4.13. From the data it can be observed that, the best collision avoidance ability for this occasion is achieved at $r_c = 0.40$ and $r_m = 0.75$. The best solution quality is obtained at $r_c = 0.15$ and $r_m = 0.40$;

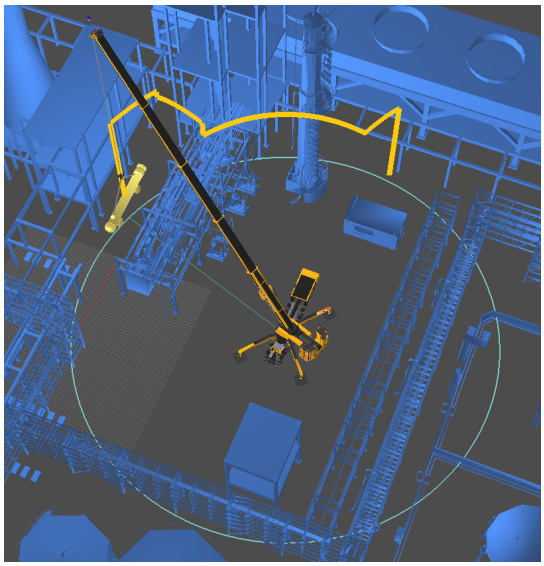
In the two experiment cases, similar patterns are observed in the topographic maps. Figure 4.12(a) and Figure 4.13(a) show that larger mutation rates helps to avoid collisions in the MSPGA search. On the other hand, the best solution qualities would be obtained in the center areas according to Figures 4.12(b) and 4.13(b). The complexity of the environment between the destinations do affect the shapes of the patterns,



(a) Path generated using the C-space strategy



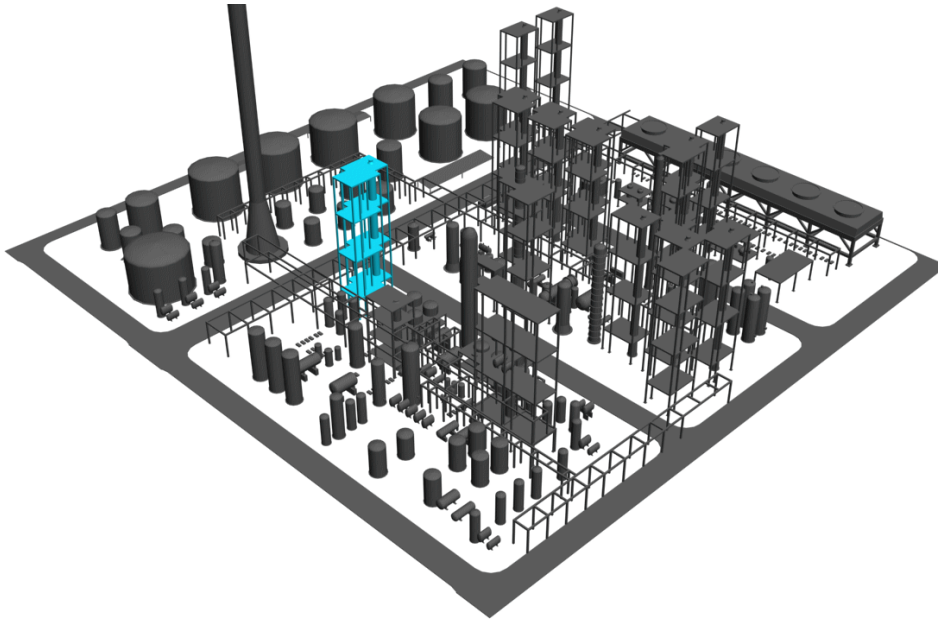
(b) Path generated using the online strategy



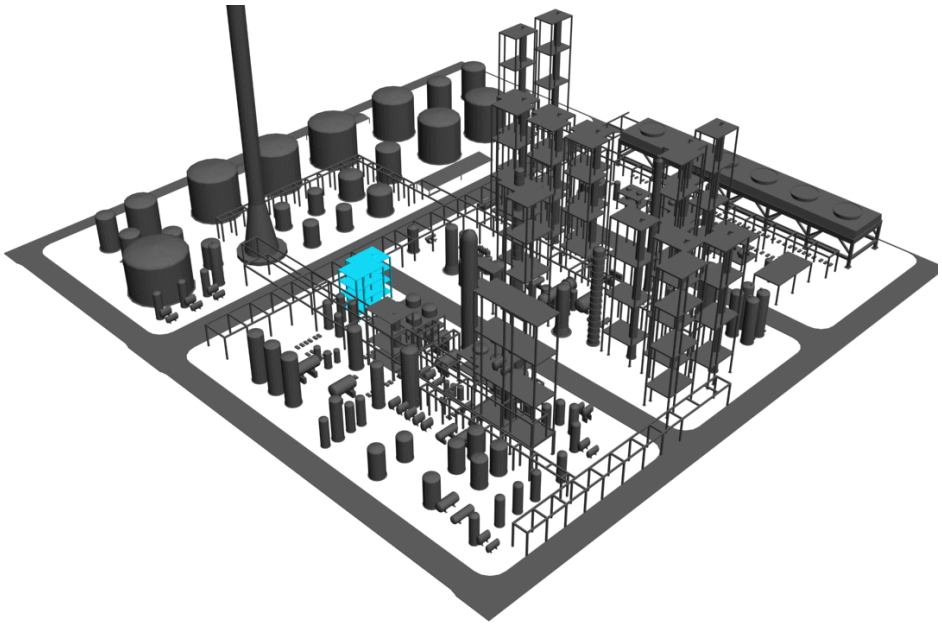
(c) Path generated using the hybrid strategy

Figure 4.10: Trajectory of the load of the result paths using the three strategies in Experiment 4.2. Green: the C-space strategy; Red: the online strategy; Yellow: the hybrid strategy

especially on the distribution of solution qualities. It may be possible that, the statistical patterns can be represented as functions of the complexity of the environment between the start and end position. Although the performance of the algorithm is



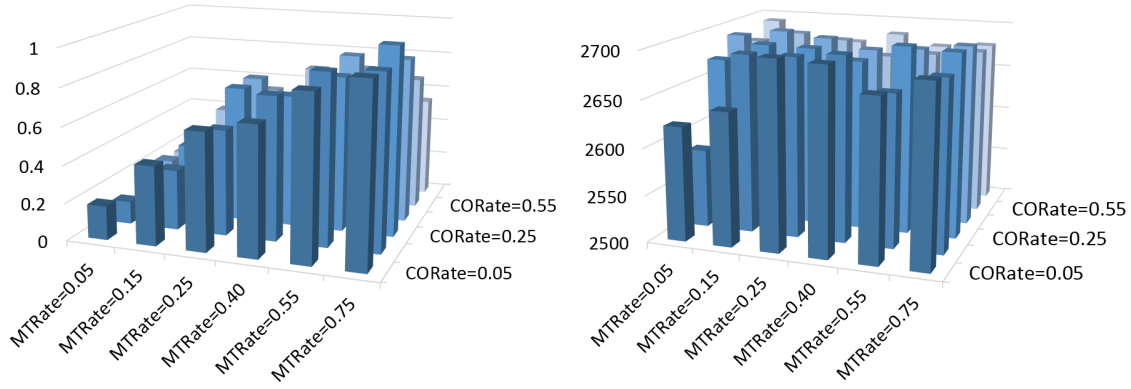
(a) Plant model for Experiment 4.3.1: Higher blue color obstacle.



(b) Plant model for Experiment 4.3.2: Lower blue color obstacle.

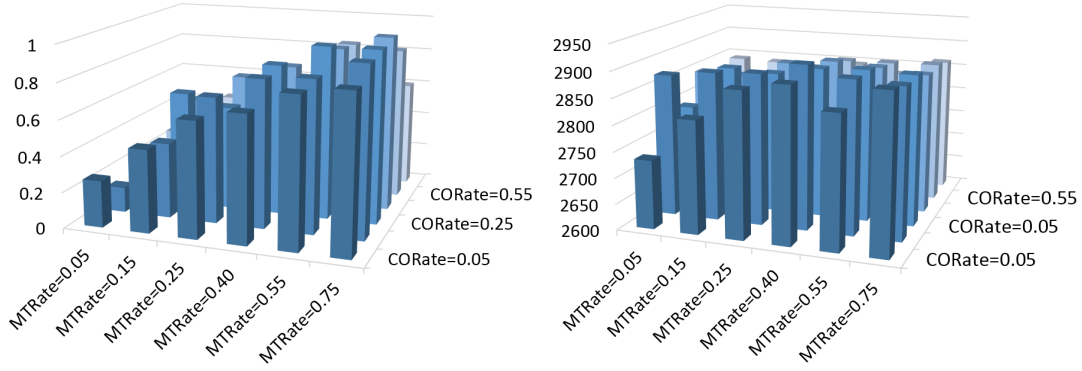
Figure 4.11: Experiment plant models

somehow dependent on its parameter design, there is a chance to predict the optimum parameter by the pre-analysis of the complexity of the plant environment. This might be a future direction of the investigation.



(a) Success rates (b) Average fitness values

Figure 4.12: Topographic maps of success rates and average fitness values different combinations of reproduction rates in Experiment 4.3.1



(a) Success rates (b) Average fitness values

Figure 4.13: Topographic maps of success rates and average fitness values different combinations of reproduction rates in Experiment 4.3.2

Table 4.15: Probabilities of finding feasible solutions under different combinations of reproductive rates for Experiment 4.3.1

	$r_m = 0.05$	$r_m = 0.15$	$r_m = 0.25$	$r_m = 0.40$	$r_m = 0.55$	$r_m = 0.75$
$r_c = 0.05$	0.18	0.42	0.62	0.68	0.86	0.94
$r_c = 0.15$	0.12	0.32	0.56	0.76	0.9	0.92
$r_c = 0.25$	0.2	0.38	0.72	0.7	0.82	1
$r_c = 0.40$	0.2	0.36	0.72	0.66	0.88	0.88
$r_c = 0.55$	0.18	0.46	0.6	0.74	0.76	0.72
$r_c = 0.75$	0.12	0.12	0.32	0.32	0.5	0.54

Table 4.16: Average fitness value for collision-free results under different combinations of reproductive rates for Experiment 4.3.1

	$r_m = 0.05$	$r_m = 0.15$	$r_m = 0.25$	$r_m = 0.40$	$r_m = 0.55$	$r_m = 0.75$
$r_c = 0.05$	2621.57	2641.26	2697.87	2695.51	2669.43	2687.2
$r_c = 0.15$	2583.56	2689.6	2690.29	2695.53	2661.03	2680.24
$r_c = 0.25$	2671.82	2691.26	2690.43	2680.13	2698.68	2695.71
$r_c = 0.40$	2690.81	2698.15	2693.12	2683.36	2687.22	2693.53
$r_c = 0.55$	2676.43	2688.1	2683.17	2668.07	2673.2	2678.88
$r_c = 0.75$	2692.9	2659.97	2673	2684.72	2673.45	2677.55

Table 4.17: Probabilities of finding feasible solutions under different combinations of reproductive rates for Experiment 4.3.2

	$r_m = 0.05$	$r_m = 0.15$	$r_m = 0.25$	$r_m = 0.40$	$r_m = 0.55$	$r_m = 0.75$
$r_c = 0.05$	0.26	0.46	0.64	0.7	0.82	0.86
$r_c = 0.15$	0.14	0.42	0.7	0.82	0.84	0.94
$r_c = 0.25$	0.26	0.64	0.58	0.84	0.96	0.96
$r_c = 0.40$	0.24	0.54	0.7	0.78	0.9	0.98
$r_c = 0.55$	0.26	0.38	0.56	0.66	0.88	0.86
$r_c = 0.75$	0.08	0.44	0.54	0.42	0.46	0.6

Table 4.18: Average fitness value for collision-free results under different combinations of reproductive rates for Experiment 4.3.2

	$r_m = 0.05$	$r_m = 0.15$	$r_m = 0.25$	$r_m = 0.40$	$r_m = 0.55$	$r_m = 0.75$
$r_c = 0.05$	2732.36	2816.53	2878.26	2894.7	2852.79	2898.74
$r_c = 0.15$	2873.98	2884.97	2889.7	2912.74	2892.72	2887.05
$r_c = 0.25$	2792.33	2876.66	2872.2	2887.83	2892.21	2888.91
$r_c = 0.40$	2845.68	2844.28	2875.95	2885.69	2879.5	2871.27
$r_c = 0.55$	2797.59	2748.11	2808	2870.67	2871.94	2874.16
$r_c = 0.75$	2841.87	2839.74	2834	2846.18	2832.1	2862.73

4.6 Summary

This chapter have proposed a new automatic path planning algorithm for single-crane lifting. A comprehensive mathematical formulation is provided for the problem and a customized MSPGA is developed to tackle the optimization problem. Compared to Ali's fitness function, the proposed fitness function shows great advantage on the success rates. In order to deal with complex environments, the image-space collision

detection introduced in Chapter 3 is combined with ASVs to perform DCD and CCD in the GA search. Finally, to further improve the computational efficiency, a hybrid C-space collision detection strategy is proposed. This strategy preserves the good solution quality of the online strategy and reduces the pre-processing time of the C-space strategy significantly.

The lifting path planning algorithm is able to handle complex plant environments and output safe lifting paths highly optimized in terms of energy cost and human conformity. The result paths are smooth and concise which are ready to be used by real cranes. Although the performance of the algorithm is somehow dependent on its parameter design, it is possible to predict the optimum parameter by the pre-analysis of the complexity of the plant environment.

Chapter 5

Dual-Crane Lifting Path Planning Using LGP-enhanced Parallel Genetic Algorithm

5.1 Introduction

Cooperative dual-crane lifting is a common strategy applied in heavy and critical lifting tasks. The intensive coordination and synchronization required in dual-crane lifting lead to difficulties in planning and executions. Safety is crucial when dealing with the irregular movements of the suspended lifting target. This chapter concerns the dual-crane lifting path planning problem in complex environments such as construction sites, petrochemical and pharmaceutical plants. It is a multi-agent path planning problem involving a closed kinematic chain formed by the suspended lifting target. The inputs to the problem include crane setup information, the pick & place locations of the lifting target, and the plant environment. The goal is to obtain well-coordinated, collision-free and short dual-crane lifting paths for complex environments.

Early exploration of the problem is in [141] where discrete searching algorithms such as hill climbing and A* are performed in the C-space of the cranes as a 2×3 manipulator. The cranes are modeled as simple sets of boxes. The 6D C-space is explicitly calculated, taking up the majority of its computation time. The work has

This chapter is based on the journal paper: P. Cai, Y. Cai, J. Zheng and I. Chandrasekaran, "Automatic path planning for dual-crane lifting in complex environments using GPU-enabled parallel genetic algorithm", to be submitted.

been further extended in [1] using a GA to optimize the candidate dual-crane paths in the C-space. The GA-based approach in [1] is able to provide highly optimized solutions. However, their method is still prohibited due to the expensive computation cost and thus restrained in simple environments and crane representations only. The latest work conducted in [2] utilizes a PRM in dual-crane lifting planning. The algorithm separates the optimization of hoisting heights from swinging and luffing operations by constraining zero tilting angle of the lifting target. Optimization of swinging and luffing operations are conducted using a PRM where the roles of the major and assistant cranes are switched in each iteration. Rule-based hoisting planning is then conducted along the swinging and luffing path by choosing optimal heights for the lifting target. Their method is able to achieve near real-time performance in simple environments. However, the problem decomposition used is not valid for lifting cases requiring tilting of the target.

This chapter first analyzes the DOFs of the dual-crane lifting problem and formulates the mathematical model. Then, the manipulation and suspension sub-systems are handled separately to determine the exact state of the dual-crane lifting system accurately. The proposed solution solves the equilibrium of the lifting target by formulating a non-linear equation system considering kinematic constraints and physical constraints such as force balancing and moment equilibrium. This chapter then discusses efficient representations of coordinations between the cranes. Based on these mathematical formulations, the MSPGA-based path planner for dual-crane lifting is then developed. The proposed planner exploits the LGP strategy to handle the multi-objective nature of the dual-crane path planning problem. The adaptive plan and the fitness function are designed to reflect the priorities of the multiple objectives and thus help the GA towards convergence. The planner avoids explicit representation of the high DOF C-space of dual-crane lifting. Instead, it decomposes the 6D C-space into two 2D C-spaces and handles the rest of the motions by online collision checks. The proposed planner employs hybrid C-spaces and TSSs to evaluate the safety factors of the dual-crane lifting paths. This chapter details on how this CCD algorithm deals with the uncertain motion of the dual-crane lifting target soundly and efficiently. Finally, this chapter presents comparisons with two previous algorithms to show the

effectiveness and efficiency of the proposed solution. The algorithm developed in this chapter, together with the ones introduced in Chapters 3 and 4 provide technology supports for the lift planner cum crane simulator system that will be presented in Chapter 6.

5.2 Mathematical Formulation

The industrial plant or construction site that the cranes are working in is considered as a complex static environment. Assumptions made to the motion of the cranes and the lifting target are similar to those stated in the Chapter 4. During the process of dual-crane lifting, the cranes, the lifting target, and the ground form a closed kinematic chain which brings high non-linearity and motion uncertainty to the path planning problem.

In order to deal with the closed kinematics chain in dual-crane lifting, The system is divided into two sub-systems: the manipulation sub-system and the suspension sub-system. The manipulation sub-system is composed of two cranes that can be represented as two open kinematic chains. The lifting target suspended from the boom tips of the cranes via slings is the suspension sub-system. A single terrain crane has 2 DOFs excluding target rotation and sling length. Thus, the manipulation sub-system has 2×2 DOFs considering the two terrain cranes. Figure 5.1 shows the 4 DOFs of the dual-crane manipulation sub-system.

Generally, each sling in the suspension sub-system has 3 variables: sling length l, yaw angle θ and tilt angle ϕ. The lifting target also has 5 variables in the suspension sub-system: anchor position $A = (A_x, A_y, A_z)$, tilt angle φ and yaw angle ϑ. However, these motion variables are not independent. Under the constraints of kinematics and equilibrium, the two sling lengths l_1 and l_2 can determine the rest of variables (details will be discussed in Section 5.4). Consequently, the suspension sub-system only has two Degrees of Freedom (DOFs): l_1 and l_2. In total, the dual-crane lifting system has $2 \times 2 + 2 = 6$ DOFs. In other words, the dual-crane lifting paths are defined in a 6D C-space, denoted as C_{dual} in this chapter. The structure of the suspension sub-system can be found in Figure 5.3(a).

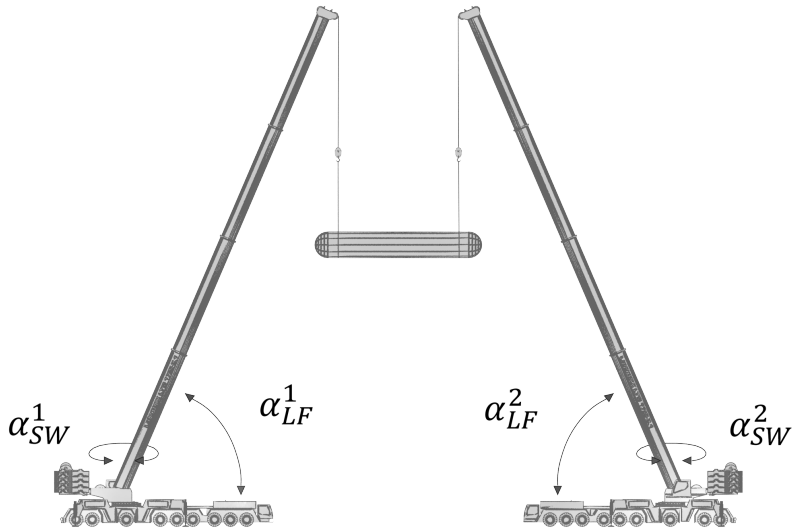


Figure 5.1: Kinematics and DOFs of the manipulation sub-system of dual-crane lifting.

Based these analyses, the mathematical representation of the optimization problem of dual-crane lifting path planning can be formulated following a similar procedure as Section 4.2.2 in Chapter 4.

In the proposed algorithm, the dual-cane lifting path is represented as a string $s_{dual} = \{O_{dual}, E_{dual}\}$, where O_{dual} is the set of nodes (dual-crane configurations) and E_{dual} represents the set of edges (internal dual-crane paths between independent nodes). Thus, the variables of the optimization problem can be written as (see Table 5.1 for explanations of the symbols):

$$s_{dual} = \{c_i\}_{i=0,1,...,L_s-1} \cup \{e_j\}_{j=0,1,...,L_s-2} \tag{5.1}$$

$$c_i = (\alpha^1_{LF}, \alpha^1_{SW}, l^1_{HS}, \alpha^2_{LF}, \alpha^2_{SW}, l^2_{HS}) \tag{5.2}$$

The edges e_j are composed by the set of dual-crane configurations determined by nodes c_j and c_{j+1} through linear interpolation of parameters. This definition of inputs will be reorganized as chromosomes in the GA search in Section 5.5. The task of the planning algorithm is to find an optimal s^*_{dual} composed of c^*_i which maximizes the evaluation function.

A metric function is defined in space C_{dual} in order to evaluate the distances. The metrics d' is defined as:

$$d'(a,b) = \sum_{i=0}^{6} r_i |a_i - b_i| \tag{5.3}$$

Table 5.1: Parameters and variables used in solution representation

Symbol	**Expression**
c_i	The ith dual-crane configuration in the string s_{dual}
e_j	The jth edge in the string s_{dual}
L_s	The length (number of configurations) of the string s_{dual}
α_{LF}^1	The luffing angle of crane 1 in degrees
α_{SW}^1	The swinging angle of crane 1 in degrees
l_{HS}^1	The hoisting length of crane 1 in centimeters
α_{LF}^2	The luffing angle of crane 2 in degrees
α_{SW}^2	The swinging angle of crane 2 in degrees
l_{HS}^2	The hoisting length of crane 2 in centimeters

Here a and b denote respectively two configurations in space C_{dual} and a_i, b_i $(i = 0, \ldots, 6)$ are the unified representation of the six parameters of a and b (Equation (5.1)). A scaling factor r_i is applied to the absolute difference value for each dimension in d'. Generally, d' measures the total number of weighted movement units of the two terrain cranes. The weights indicate the energy cost of the correspondent crane when unit movement is conducted. When r_i is set as 1 for all i, function d' is equivalent to the L1 norm defined on 6D vectors.

Now the evaluation function of a string s_{dual} in the solution space S_{dual} can be expressed as (see Table 5.2 for explanations of the symbols):

$$F(s_{dual}) = \lambda \left(1 + \frac{\lambda}{d(s_{dual})}\right) \tag{5.4}$$

where

$$d(s_{dual}) = \sum_{i=0}^{L_s-2} d'(c_i, c_{i+1}) \tag{5.5}$$

$$s_{dual} \in S_{dual}, c_i \in C_{dual} \quad and \quad i = 0, \ldots, L_s - 1 \tag{5.6}$$

Then the maximization problem for the dual-crane lifting path planning scenario can be accordingly written as:

Table 5.2: Parameters and variables in the objective function

Symbol	**Expression**
$F(s)$	The evaluation value of string s_{dual}
$d(s)$	The distance cost in string s_{dual}
c_j	The jth configuration in string s_{dual}
λ, λ'	Constant scaling factors

$$
\begin{aligned}
\max \quad & F(s_{dual}) \\
s.t. \quad & n_{node}(s_{dual}) = 0,\ s_{dual} \in S_{dual} && (5.7)\\
& n_{edge}(s_{dual}) = 0,\ s_{dual} \in S_{dual} && (5.8)\\
& m_{node}(s_{dual}) = 0,\ s_{dual} \in S_{dual} && (5.9)\\
& m_{edge}(s_{dual}) = 0,\ s_{dual} \in S_{dual} && (5.10)\\
& \underline{B_{dual}} \leq c_i \leq \overline{B_{dual}}, && (5.11)\\
& i = 0, 1, \ldots, L_s - 1 && (5.12)\\
\text{where} \quad & n_{node}(s_{dual}) = \sum_{i=0}^{L_s-1} \delta(c_i) && (5.13)\\
& n_{edge}(s_{dual}) = \sum_{i=0}^{L_s-2} \delta(e_i) && (5.14)\\
& m_{node}(s_{dual}) = \sum_{i=0}^{L_s-1} \sigma(c_i) && (5.15)\\
& m_{edge}(s_{dual}) = \sum_{i=0}^{L_s-2} \sigma(e_i) && (5.16)\\
& \delta(c_i) \in \{0, 1\}, i = 0, 1 \ldots, L_s - 1 && (5.17)\\
& \delta(e_i) \in \{0, 1\}, i = 0, 1 \ldots, L_s - 2 && (5.18)\\
& \sigma(e_i) \in \{0, 1\}, i = 0, 1 \ldots, L_s - 2 && (5.19)\\
& \sigma(c_i) \in \{0, 1\}, i = 0, 1 \ldots, L_s - 1 && (5.20)
\end{aligned}
$$

The goal function of this maximization problem is the evaluation function defined in Equation (5.4). Differed from the single-crane problem, the dual-crane problem

has five constraints: discrete collision constraint (Equation (5.7)), continuous collision constraint (Equation (5.8)), coordination constraint (Equation (5.9)), continuous coordination constraint (Equation (5.10)) and DOF limit constraint (Equation (5.11)). $\delta(c_i)$ and $\delta(e_i)$ represent the collision detection results of elements c_i and e_i in string s_{dual}. $\sigma(c_i)$ and $\sigma(e_i)$ represent the coordination test results of elements c_i and e_i in string s_{dual}. The two types of collision constraints require that there are no collision violations in the node configurations (discrete) or the edge paths (continuous). Details of the collision detection computations will be discussed in Section 5.6. The two coordination constraints require the cranes to be coordinated in each node and during the movement along the edges (continuous). Details about the coordination of cranes will be discussed in Section 5.4.2. $\underline{B_{dual}}$ and $\overline{B_{dual}}$ stand for the lower and upper bound values for the 6 DOFs in the dual-crane configurations. $\underline{B_{dual}}$ and $\overline{B_{dual}}$ are empirically set as $(0, 0, 100, 0, 0, 100)$ and $(82, 360, 7000, 82, 360, 7000)$. The optimal solution s^*_{dual} of the maximization problem is a dual-crane lifting path which is well-coordinated, collision-free and optimized in energy cost.

5.3 The Manipulation Sub-system

The manipulation sub-system includes components of the terrain cranes except the sling and hook. The position and orientation of these components are fully determined by the kinematics as shown in Figure 5.2.

O_i in Figure 5.2(b) stands for the ith node in the manipulation sub-system. As shown in Figure 5.2(a), the nodes represent different components of the terrain cranes. O_0 is the center of the crane body. O_1 stands for the center of the rotational plate supporting the cockpit and the upper structures. O_2 is the rotational anchor of the main boom which usually has a cylindrical shape. O_3 represents the tip of the boom. The connection line of O_2 and O_3 is parallel to the boom. O_4 stands for the anchor where the sling is connected with the boom. L_i is the ith link in the manipulation sub-system that connects node O_{i-1} and O_i. The manipulation sub-system has 3 DOFs as shown in Figure 5.2(c). α_{BD}, α_{SW} and α_{LF} represent the angle of body rotation, cockpit swinging, and boom luffing respectively. Among them, α_{LF} is the angle from the x-y plane while the rest two are angles along the z-axis.

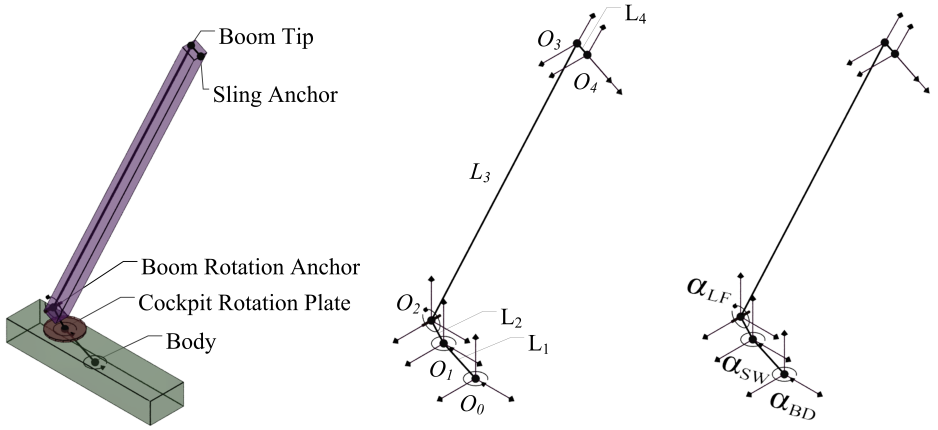


Figure 5.2: The manipulation sub-system of dual-crane lifting: (a) names of the nodes in terrain cranes; (b) joints and links in the manipulation sub-system; (c) DOFs of the manipulation sub-system.

Solving the manipulation sub-system is to obtain the position of the end effector O_4 from the kinematic structure. The sling anchor O_4 will serve as the input for the dual-crane suspension sub-system. The position of O_4 can be calculated through the following equation:

$$\begin{aligned} O_4 &= O_0 + \mathbf{R_z}(\alpha_{BD}) \\ &\quad (L_1 + \mathbf{R_z}(\alpha_{SW})\,(L2 + \mathbf{R_x}(\alpha_{LF})\,(L3 + L4))) \end{aligned} \tag{5.21}$$

where $\mathbf{R_z}(\alpha)$ stands for the matrix representing the rotation along the z axis with angle α. Similarly, $\mathbf{R_x}(\alpha)$ stands for the rotation matrix along the x axis.

5.4 The Suspension Sub-system

The suspension sub-system of dual-crane lifting considers the load suspended on the sling anchors through the slings and riggings. The positions of the sling anchors are acquired from solving the manipulation sub-systems. Components of the suspension sub-system include the slings and hooks of the cranes, the lifting target and rigging components connecting them.

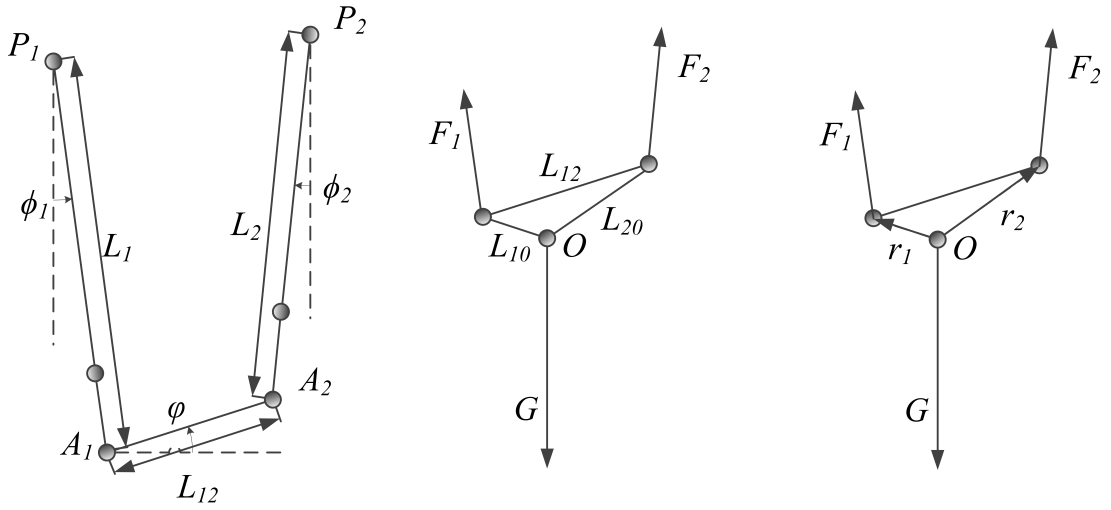
5.4.1 Solving the dual-crane suspension sub-system

In order to solve the suspension sub-system, the weight of the slings and rigging components are ignored due to they are much smaller than the weight of the lifting target in heavy lifting. Moreover, in an equilibrium state, the mass center of the lifting target, and rigging anchor points must lie in a single vertical plane. As a result, the suspension sub-system is simplified as a lifting target suspended from the sling anchors through two weightless links in a 2D plane. Figure 5.3(a) shows the kinematic structure of the suspension sub-system. The 4 nodes in the suspension sub-system include P_1 and P_1 which are the sling anchors, and A_1 and A_2 which are the attach anchors on the lifting target. The 5 variables in the system include ϕ_1 and ϕ_2 that are the angles between the slings and the z-axis, L_1 and L_2 (the lengths of the slings) and φ (the tilt angle of the lifting target). Because of the 2 hinge joints at A_1 and A_2, the DOF of the suspension sub-system is reduced to 3.

The lifting target is simplified as a triangle structure containing the two attach anchors A_1, A_2 and the mass center O (Figure 5.3(b)). The length between the two attach anchors on the lifting target is denoted as L_{12}. The distances from the attach anchors to the mass center are denoted as L_{10} and L_{20}.

The positions of the mass center and attach anchors cannot be determined by the kinematic alone since the sub-system has 3 DOFs but only has 2 parameters L_1 and L_2 to manipulate. To compute all the parameters, it is also necessary to consider the equilibrium of forces and moments of the lifting target under the effect of gravity and the sling forces.

In order to formulate the balancing equation of the suspension sub-system, the following parameters are introduced as variables of the equations: $\phi_1, \phi_2, \varphi, f_1, f_2$ and O. f_1 and f_2 are the magnitudes of forces F_1 and F_2. Other variables are the same as defined previously. The relationships among the variables are defined by 4 constraints: kinematics of the suspension sub-system, shape preservation of the lifting target triangle, equilibriums of forces and moments. The equation system is written as below accordingly:



(a) (b) (c)

Figure 5.3: Kinematics, forces and moments of the suspension sub-system in dual-crane lifting.

$$P_1 + L_1 \begin{pmatrix} sin(\phi_1) \\ -cos(\phi_1) \end{pmatrix} + L_{12} \begin{pmatrix} cos(\varphi) \\ sin(\varphi) \end{pmatrix} = P_2 + L_2 \begin{pmatrix} -sin(\phi_2) \\ -cos(\phi_2) \end{pmatrix} \tag{5.22}$$

$$||P_1 + L_1 \begin{pmatrix} sin(\phi_1) \\ -cos(\phi_1) \end{pmatrix} - O|| = L_{10} \tag{5.23}$$

$$||P_2 + L_2 \begin{pmatrix} -sin(\phi_2) \\ -cos(\phi_2) \end{pmatrix} - O|| = L_{20} \tag{5.24}$$

$$F_1 + F_2 = G \tag{5.25}$$

$$r_1 \times F_1 + r_2 \times F_2 = 0 \tag{5.26}$$

Equation (5.22) represents the kinematic constraint. It states that the attach anchor A_2 on the lifting target has to be aligned with the tip of the second sling. Equations (5.23) and (5.24) describe the shape of the triangle in the lifting target. They require the mass center to be in a fixed relative position in the lifting target. Equation (5.25) containing 2 dimensions represents the equilibrium of gravity force and the sling forces. Equation (5.26) states the equilibrium of moments on the mass center of the lifting target applied by the gravity and the sling forces. These nonlinear equations are solved using the "hybrids" solver in the GNU Scientific Library [147] which uses the finite difference approximation of Jacobian functions to guide the direction of searches.

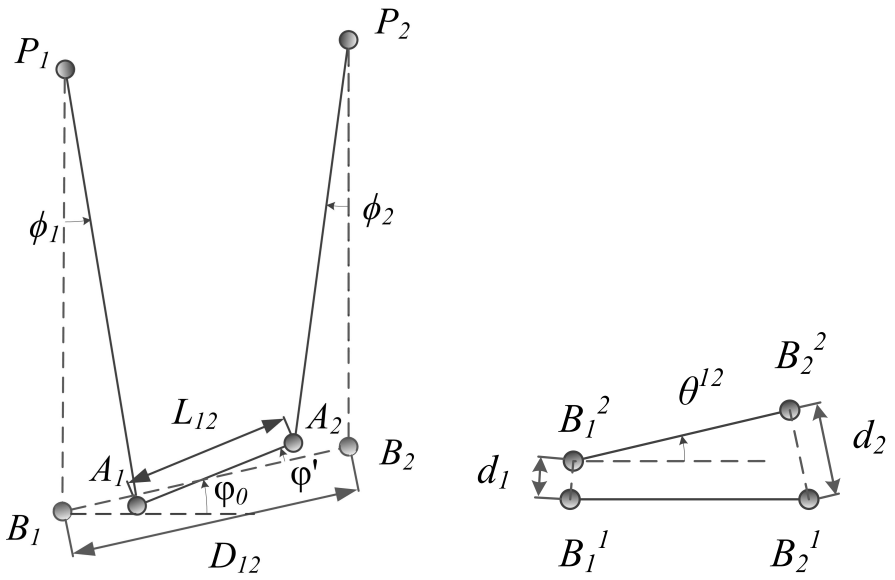


Figure 5.4: The two types of coordination of the suspension sub-system: (a) node coordination; (b) continuous coordination.

5.4.2 Coordination of cranes

The purpose of using dual cranes for lifting is to share the load of the lifting target on the two cranes. The capacities of the two cranes are usually chosen as slightly larger (with a safety factor) than half of the load to handle the extra forces caused by sling tilting. However, the forces or load on the sling increase non-linearly with the tilt angle of the slings. Thus in practice, a safety threshold, denoted as ϕ_{max}, is set to the tilt angle of the slings. When the absolute values of ϕ_1 and ϕ_2 are smaller than ϕ_{max}, the dual-crane configuration is called "coordinated" in the equilibrium state. Otherwise, the configuration is not properly coordinated. There are two possible ways to check coordination during the lifting path planning. The first method is to solve the suspension equation system and check ϕ_1 and ϕ_2 against ϕ_{max}. This method is accurate but computationally prohibited due to the high computational cost to solve the 7D non-linear equation system. The alternative way is to approximate ϕ_1 and ϕ_2 with some easy-to-get value (details see below).

For the approximation, two points B_1 and B_2 are introduced as estimated attach anchors. As shown in Figure 5.4(a), B_1 and B_2 stand respectively for the positions of

A_1 and A_2 when both ϕ_1 and ϕ_2 are set to zero. Thus there are two relationships:

$$|A_1B_1| = L_1 \sin(\phi_1) = L_1\phi_1 + O(\phi_1^3) \quad (5.27)$$

$$|A_2B_2| = L_2 \sin(\phi_2) = L_2\phi_2 + O(\phi_2^3) \quad (5.28)$$

For small ϕ_1 and ϕ_2, Equation (5.28) is further written as:

$$\begin{aligned} |A_1B_1| + |A_2B_2| &= L_1|\phi_1| + L_2|\phi_2| + O(|\phi_1|^3) + O(|\phi_2|^3) \\ &\approx L_1|\phi_1| + L_2|\phi_2| \end{aligned} \quad (5.29)$$

The value $L_1|\phi_1| + L_2|\phi_2|$ can thus be used to monitor the coordination for dual-crane lifting as indicated in the following equation:

$$L_1|\phi_1| + L_2|\phi_2| < min(L_1, L_2)\phi_{max} \quad (5.30)$$

It is a sufficient condition for coordination because it always has:

$$min(L_1, L_2)(|\phi_1| + |\phi_2|) < L_1|\phi_1| + L_2|\phi_2| \quad (5.31)$$

Therefore, satisfying Equation (5.30) can make sure that $|\phi_1| + |\phi_2| < \phi_{max}$. However, this value is still dependent on ϕ_1 and ϕ_2 which can only be acquired through solving the suspension sub-system. Evaluating these values during lifting path planning is computationally forbidden. Therefore, more steps are used to simplify the representation. Note that when we translate A_1A_2 so that A_2 coincide with B_2, a triangle is formed by A_1, B_1 and B_2 with the length of the bottom edge to be $L = L_1|\phi_1| + L_2|\phi_2|$. By performing projections of A_1A_2 and $B_1A_1 + A_2B_2$ on B_1B_2, we can get the following relationships:

$$Lcos(\phi_0) = D_{12} - L_{12}cos(\phi') \quad (5.32)$$

$$Lcos(\phi_0) = D_{12} - \frac{D_{12}^2 + L_{12}^2 - L^2}{2D_{12}} \quad (5.33)$$

$$L = \frac{D_{12}}{2cos(\phi_0)} - \frac{L_{12}^2}{2D_{12}cos(\phi_0)} + \frac{L^2}{2D_{12}cos(\phi_0)} \quad (5.34)$$

Here ϕ_0 is the tilting angle of B_1B_2 and ϕ' is the angle between A_1A_2 and B_1B_2 (see Figure 5.4(a)). L_{12} and D_{12} refer to the length of A_1A_2 and B_1B_2 respectively. Since

it always has:

$$L < D_{12}cos(\phi_0) \tag{5.35}$$

By substituting $D_{12}cos(\phi_0)$ into Equation (5.34), we get:

$$L < \frac{D_{12}}{2cos(\phi_0)} - \frac{L_{12}^2}{2D_{12}cos(\phi_0)} + \frac{L}{2} \tag{5.36}$$

$$L < \frac{D_{12}}{cos(\phi_0)} - \frac{L_{12}^2}{D_{12}cos(\phi_0)} = \frac{D_{12}^2 - L_{12}^2}{D_{12}cos(\phi_0)} \tag{5.37}$$

In the proposed planner, the value $e = |D_{12}^2 - L_{12}^2| / (D_{12}cos(\phi_0))$ is used to approximate $L_1|\phi_1| + L_2|\phi_2|$. The final coordination constraint is formulated as:

$$\frac{|D_{12}^2 - L_{12}^2|}{D_{12}cos(\phi_0)} < min(L_1, L_2)\phi_{max} \tag{5.38}$$

When this inequality is met, combining Equations (5.30), (5.31) and (5.37), it can be easily seen that $|\phi_1|$ and $|\phi_2|$ will be smaller than ϕ_{max}. This proves that the simplified constraint formulated in Equation (5.38) is still a sufficient condition for coordination.

When the cranes are conducting the output path of lifting planning, there also exists possibility of violating the dual-crane coordination when moving from one node configuration to another. Thus in lifting path planning, the coordination during the movements along the edges (continuous coordination) shall also be considered. The continuous coordination is implemented through constraining the distance traveled by the estimated attach anchors and the angle between the lifting target direction in neighboring steps. The continuous coordination constraint is written as:

$$||(B_1^1 - B_1^2)_{xy}|| + ||(B_2^1 - B_2^2)_{xy}|| \leqslant \lambda(L_1 + L_2)\phi_{max} \tag{5.39}$$

$$|(B_1^2 - B_1^1)_z - (B_2^2 - B_2^1)_z| \leqslant \lambda'(L_1 + L_2)\phi_{max} \tag{5.40}$$

As illustrated in Figure 5.4(b), B_1^1 and B_2^1 are the estimated attach anchors on the lifting target in time step 1, and B_1^2 and B_2^2 are the estimated attach anchors in the neighboring time step 2. Equation (5.39) states that the distance traveled by the two

attach anchors has to be within a small threshold related to ϕ_{max}. The scaling factor λ is used to control the tightness of the threshold. Equation (5.40) constrains the tilting of the lifting target in a threshold related to ϕ_{max}. Synchronized hoisting which does not affect the coordination of the cranes is not constrained in this equation. The continuous coordination constraint helps to control the sling angles in a small range during the whole path. Even though it is not a sufficient condition, the parameters can be fine-tuned with the help of the suspension sub-system solver introduced in Section 5.4.1 to produce strictly coordinated paths.

5.5 MSPGA-based Path Planner for Dual-crane Lifting

This section presents the customized and LGP-enhanced MSPGA for the dual-crane lifting path planning problem. Collision avoidance of the planner is based on the hybrid C-space collision strategy as introduced in Chapter 4 with approximated swept volumes for the lifting target and multi-level depth maps introduced in Chapter 3. The collision detection module will be further discussed in Section 5.6.

A chromosome in the MSPGA-based path planner for dual-crane lifting is defined as the array of dual-crane configurations (genes). The chromosome is composed of all the node configurations c_i $(i = 0, \ldots, L_s - 1)$ from the solution s_{dual} in Equation (5.1) in Section 5.2. The population is a fixed-size set of chromosomes carrying different dual-crane lifting paths (candidates) to evolve in the GA process.

5.5.1 Framework of the LGP-enhanced MSPGA

The MSPGA-based path planner takes the inputs from the simulation system and output optimized lifting paths through the MSPGA evolutions. The start and end configurations predefined by the user are the major inputs to the MSPGA-based path planner. The parameters such as string length, population size and the rates of adaptive operators are also passed to the planner as inputs. Moreover, the kinematics and hybrid C-spaces of the cranes are input into the planner for coordination and collision checking. Figure 5.5 is a brief denotation of the work flow.

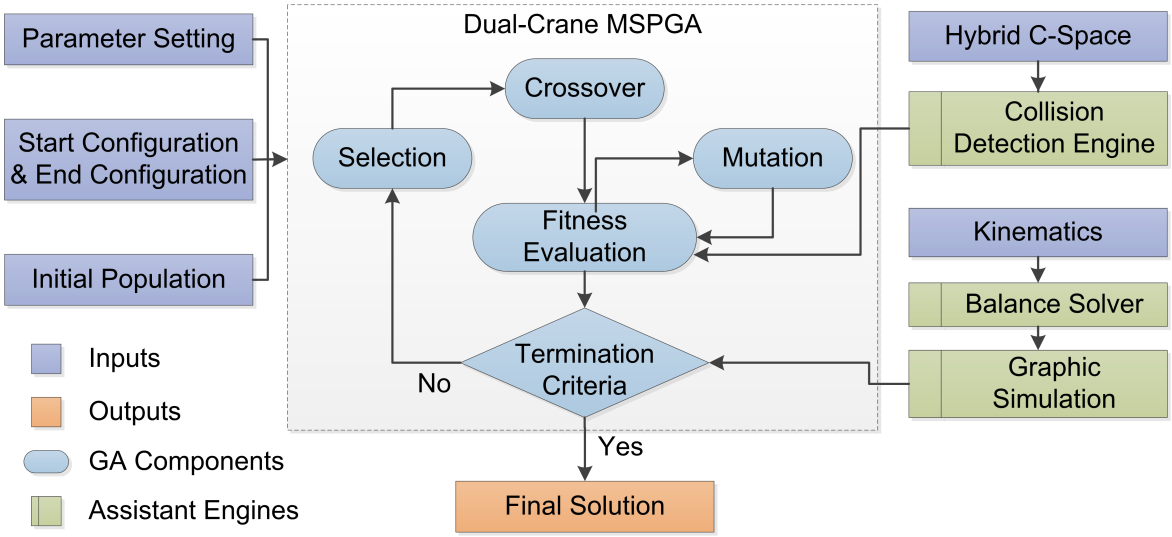


Figure 5.5: Workflow of the MSPGA-based path planner for dual-crane lifting.

5.5.2 Initialization

The search space of path planning for dual-crane lifting is much larger than that for single-crane lifting. Good initialization of paths can greatly increase the performance of GA. The initialization strategy applied in the MSPGA-based path planner for dual-crane lifting includes two parts. The major part of the population is generated through randomly selecting parameters within DOF constraints. For the rest of the population, swinging angles and luffing angles are generated through interpolating the start and end values. The hoisting values for genes are determined by choosing a random hoisting height same for both the major crane and assistant crane. The sling lengths are acquired through subtracting the maximum hoist height (related to luffing angle) with the selected hoist height. These candidates are called "seeds" fed into the population. These seeds are initial guesses of how the good solutions should look like. Although the seeds are usually not feasible solutions, they are more likely to provide good building blocks through mutation (see Section 5.5.3.2).

5.5.3 Dual-crane adaptive plan

This section focuses on the fitness function and adaptive operators of the MSPGA-based path planner for dual-crane lifting. These components of the planner have

Table 5.3: The dual-crane initialization strategy for the major portion in the population

Configuration	**Strategy**
c_1	Randomly generate hoist height; Keep other parameters as the start configuration
$c_2 \sim c_{L_s-3}$	Randomly generate parameters
c_{L_s-2}	Randomly generate hoist height; Keep other parameters as the end configuration

Table 5.4: The dual-crane initialization strategy for the seeds in the population

Parameter	**Strategy**
$\alpha^1_{SW}\&\alpha^2_{SW}$	Interpolate start and end swinging angles
$\alpha^1_{LF}\&\alpha^2_{LF}$	Interpolate start and end luffing angles
$l^1_{HS}\&l^2_{HS}$	Randomly choose hoisting height

encoded the multiple constraints in the optimization problem (see Equation (5.2)). The dual-crane fitness function and crossover operator are prioritized following the LGP principle. A coordination mutation operator is introduced to help supply well-coordinated genes.

5.5.3.1 Fitness function

Similar to the single-crane scenario, the MSPGA-based path planner for dual-crane lifting uses a proportional selection scheme where the survival chance for a string is proportional to its fitness value. The elitism strategy is also applied in order to preserve the "fittest" chromosomes. The fitness function used in the dual-crane planner is a combination of the objective function (see Section 5.4) and the hard constraints in the optimization problem (Equation (5.2)). Through this way, the multi-constraint optimization is transferred to a multi-objective optimization problem without constraint. In order to tackle the multi-objective problem, priorities are assigned to different objectives. The priorities are given from high to low as: coordination, continuous coordination, collision avoidance and continuous collision avoidance. Based on the same though as for the single-crane scenario, the fitness function is designed as a step function to differ strings in different stages. This dual-crane fitness function reflects the

Table 5.5: Parameters and variables used in the fitness function design

Symbol	**Expression**
$f(s^i_{dual})$	The fitness value of string s^i_{dual}
s^i_{dual}	The ith dual-crane string in the population
m_i	The violation number
n_i	The collision violation number
L_p	The size of population
mo_i	The coordination violation number of the nodes in s^i_{dual}
me_i	The continuous coordination violation number of the edges in s^i_{dual}
no_i	The collision violation number of nodes in string s^i_{dual}
nf_i	The continuous collision violation number of boom, cockpit and counterweight in string s^i_{dual}
nl_i	The continuous collision violation number of the lifting target in string s^i_{dual}

priority of the multiple constraints. The fitness function for a given chromosome s^i_{dual} in the population P_{dual} is defined as (see Table 5.5 for explanations of the symbols):

$$f(s^i_{dual}) = \begin{cases} \lambda_1/m_i & \text{if } m_i > 0 \\ \lambda_1(1+1/n_i) & \text{if } m_i = 0, n_i > 0 \\ \lambda_1(2+\lambda_1/d_i) & \text{if } m_i = 0, n_i = 0 \end{cases} \tag{5.41}$$

where

$$m_i = mo_i + me_i \tag{5.42}$$

$$n_i = no_i + nf_i + nt_i \tag{5.43}$$

$$i = 0, 1, 2, \ldots, L_p - 1 \tag{5.44}$$

$m_i > 0$ stands for the cranes not well coordinated at the some dual-crane configurations in the string or during the movement to neighboring configurations. When $n_i > 0$, some nodes or edges in string s_i are colliding with the environment. The fitness value lies in $(0, \lambda_1]$ when $m_i > 0$ and $(\lambda_1, 2\lambda_1]$ when $m_i = 0$ and $n_i > 0$. For the cases when $m_i = 0$ and $n_i = 0$, the fitness values are larger than $2\lambda_1$. This incremental stepwise function ensures that the strings violating higher priority constraints have lower

survival chances. The evolutionary search of GA will minimize the violation numbers from high priority to low priority. When all the constraints are met, the evolution will optimize the motion cost of the string.

5.5.3.2 Adaptive operators

The increase in the number of constraints makes the solutions much harder to find in the dual-crane lifting scenario. Thus, the success rate relies heavily on the ability of the crossover and mutation operators to provide better offspring from parents.

The MSPGA-based path planner for dual-crane lifting uses a parameter-based crossover strategy. The crossover happens with a rate r_c for each pair of parent strings in the mating pool. For each location c_i in the offspring string, the constraint violation numbers of the parent genes are compared following roughly the priority defined in Section 5.5.3.1. If one parent gene has a lower violation number for a higher priority constraint, the parent gene is picked as the offspring gene. The purpose of this strategy is to help GA enter the feasible (collision-free) space through eliminating configurations which violate the current highest priority constraint from the population. The flow of the crossover operator is illustrated in Algorithm 5.1.

Three types of mutations are used in the dual-crane planner: bitwise mutation, smoothing mutation and coordination mutation. The bitwise mutation alters bits (genes) in the population with a mutation rate r_m. A random variable in the gene is selected to add a real value within a given range. the mutation range is set larger when the coordination and collision violation numbers are greater than 0. In the smoothing mutation, genes are picked from the population with the mutation rate and one of the variables is changed as a random convex combination of those in the neighboring genes. The smoothing mutation helps the genes to be coordinated between the cranes. The coordination mutation takes a selected chromosome with the mutation rate and replaces the invalid dual-crane configurations with new coordinated configurations. It supplies the population with coordinated configurations through forcing selected genes to be coordinated.

The MSPGA-based path planner for dual-crane lifting also uses adaptive rates in all the three types of mutations. the mutation rates for a chromosome are formulated

Algorithm 5.1 Pseudo-code of the crossover strategy in the MSPGA-based path planner for dual-crane lifting

for all String s^i_{dual} in the population P_{dual} **do**
  for all Gene c_j in string s^i_{dual} **do**
    Load parent genes c^1_j and c^2_j from the mating pool;
    Penalty $g_1 \leftarrow \sigma(c^1_j)$, penalty $g_2 \leftarrow \sigma(c^2_j)$;
    if $g_1 > g_2$ **then**
      $c_j \leftarrow c^1_j$;
    else if $g_1 < g_2$ **then**
      $c_j \leftarrow c^2_j$;
    else
      Remain undetermined;
    end if
    Penalty $g_1 \leftarrow \delta(c^1_j)$, penalty $g_2 \leftarrow \delta(c^2_j)$;
    if $g_1 > g_2$ **then**
      $c_j \leftarrow c^1_j$;
    else if $g_1 < g_2$ **then**
      $c_j \leftarrow c^2_j$;
    else
      Remain undetermined;
    end if
    $g_1 \leftarrow \sigma(e^1_{j-1}) + \sigma(e^1_j)$, $g_2 \leftarrow \sigma(e^2_{j-1}) + \sigma(e^2_j)$;
    if $g_1 > g_2$ **then**
      $c_j \leftarrow c^1_j$;
    else if $g_1 < g_2$ **then**
      $c_j \leftarrow c^2_j$;
    else
      Remain undetermined;
    end if
    $g_1 \leftarrow \delta(e^1_{j-1}) + \delta(e^1_j)$, $g_2 \leftarrow \delta(e^2_{j-1}) + \delta(e^2_j)$;
    if $g_1 > g_2$ **then**
      $c_j \leftarrow c^1_j$;
    else if $g_1 < g_2$ **then**
      $c_j \leftarrow c^2_j$;
    else
      Remain undetermined;
    end if

as (see Table 5.6 for the symbols):

$$r_m(s_{dual}) = \begin{cases} \overline{r_m} + (\overline{f} - f(s_{dual}))/\overline{f}, & \text{if } f(s_{dual}) < \overline{f} \\ \overline{r_m}, & \text{if } f(s_{dual}) \geq \overline{f} \end{cases} \tag{5.45}$$

Algorithm 5.1 (continued) Pseudo-code of the crossover strategy in the MSPGA-based path planner for dual-crane lifting

```
        g1 ← d(c_j^1, c_{j+1}^1), g2 ← d(c_j^2, c_{j+1}^2);
        if g1 > g2 then
            c_j ← c_j^1;
        else if g1 < g2 then
            c_j ← c_j^2;
        else
            Remain undetermined;
        end if
        if Undetermined then
            c_j ← RandomChoose(c_j^1, c_j^2)
        end if
    end for
end for
```

Table 5.6: Parameters and variables in adaptive mutation rates

Symbol	Expression
$r_m(s_{dual})$	The mutation rate of string s_{dual}
$\overline{r_m}$	The basic mutation rate
$\overline{f}$	The average fitness value in the population
$f(s_{dual})$	The fitness value of string s_{dual}

The bit-wise mutation strategy used in the proposed algorithm is shown in Table 5.7. Similar to the single-crane scenario, the mutation only alters the hoisting height (sling lengths of the cranes) for c_1 and c_{L_s-2} in each chromosome. Mutation scales of well-coordinated or collision-free genes are kept small in order to preserve the good features. The bad genes are disturbed largely for exploring the solution space.

The smoothing mutation is done only for collision-free strings. It smooths the candidate paths by forcing the selected variable in a gene to be the interpolated value of the corresponding value in its neighbors. The procedure of the smoothing mutation is illustrated in Algorithm 5.2.

In the coordination mutation, coordinated dual-crane configurations are generated from the selected genes with the original location and orientation of the lifting target kept. The workflow of the coordination mutation is given in Algorithm 5.3.

Table 5.7: The bitwise mutation strategy used in the dual-crane planner

Configuration	Case	Strategy
c_1 & c_{L_s-2}	$\delta(c_i)+\delta(e_i)=0$	Alter hoisting height in smaller scale
	$\delta(c_i)+\delta(e_i)>0$	Alter hoisting height in larger scale
$c_2 \sim c_{L_s-3}$	$\sigma(c_i)+\sigma(e_i)=0$ or $\delta(c_i)+\delta(e_i)=0$	Alter any configuration parameter in smaller scale
	$\sigma(c_i)+\sigma(e_i)>0$ and $\delta(c_i)+\delta(e_i)>0$	Alter any configuration parameter in larger scale

Algorithm 5.2 Pseudo-code of the smoothing mutation strategy in the MSPGA-based path planner for dual-crane lifting

for all String s^i_{dual} in the population P_{dual} **do**
  if $n_i = 0$ **then**
    for all Gene c_j in string s^i_{dual} **do**
      Randomly choose variable a^j_k for mutation;
      $\lambda \leftarrow Rand(0,1)$;
      $a^j_k \leftarrow \lambda a^j_{k-1} + (1-\lambda)a^j_{k+1}$
    end for
  end if
end for

Algorithm 5.3 Pseudo-code of the coordination mutation strategy in the MSPGA-based path planner for dual-crane lifting

for all s^i_{dual} in P_{dual} **do**
  if $m_i > 0$ **then** ▷ Coordination constraints violated
    for all c_j in s^i_{dual} **do**
      if $\sigma(c_j) = 1$ **then** ▷ Node Coordination Violated
        $Dir \leftarrow Normalize(B_2 - B_1)$;
        $L \leftarrow ||B_2 - B_1||$;
        $d \leftarrow (L - L_{12})/2$;
        $B'_1 \leftarrow B_1 + d(Dir)$, $B'_2 \leftarrow B_2 - d(Dir)$
        $c_j \leftarrow (IK(B'_1), IK(B'_2))$; ▷ IK: Inverse kinematics
      end if
    end for
  end if
end for

5.6 Collision Avoidance

The collision avoidance in dual-crane lifting path planning considers both discrete and continuous collision types. Because the string length required in dual-crane lifting is much larger than in single-crane lifting, the computational load is also much heavier, especially for collision checks. Therefore, the hybrid C-space strategy proposed in Chapter 4 is also utilized here to reduce the collision detection time. In the pre-processing stage, the hybrid C-spaces of both cranes are generated and stored in the GPU memory. Collision check using the hybrid C-space is exactly as represented in Chapter 4. When the MSPGA evaluates the violation of discrete collision constraint of candidate genes, it looks up the boolean collision information from the hybrid C-space for each crane using the α_{SW}s and α_{LF}s defined in the gene being evaluated. The dual-crane gene is marked as colliding if the boolean value for any of the cranes indicates a collision. The continuous collision checks for booms, cockpit and counter-weight are also performed using the hybrid C-space. Edges in candidate strings are mapped into the hybrid C-spaces of the cranes. Collision information is checked along the mapped paths in the hybrid C-spaces. If a collision is found in any of the mapped paths for the two cranes, the edge is marked as colliding.

Collision detection of the lifting target for dual-crane lifting path planning is challenging because of the nonlinearity of the suspension sub-system. Using the exact position of the lifting target for collision check is impractical due to the high computational load. Thus, TSSs are proposed as the representations of swept volumes of the lifting target. A TSS is a volume generated by a sphere sweeping through a triangle. The TSS can also be defined as the volume dialect from the core triangle with a uniform radius. The concept of TSS is inspired from the PSS, LSS and RSS [38, 148].

The method first builds an LSS for the lifting target where the length of the core line $A'_1A'_2$ is d_{12}. The radius r of the LSS is obtained from the OBB of the lifting target such that the LSS covers the whole OBB. As the slings in the suspension sub-system are only allowed to move within a threshold ϕ_{max}, position of A'_1 and A'_2 are constraint in the neighborhood of the approximated attach anchors B_1 and B_2 (Figure 5.6(a)).

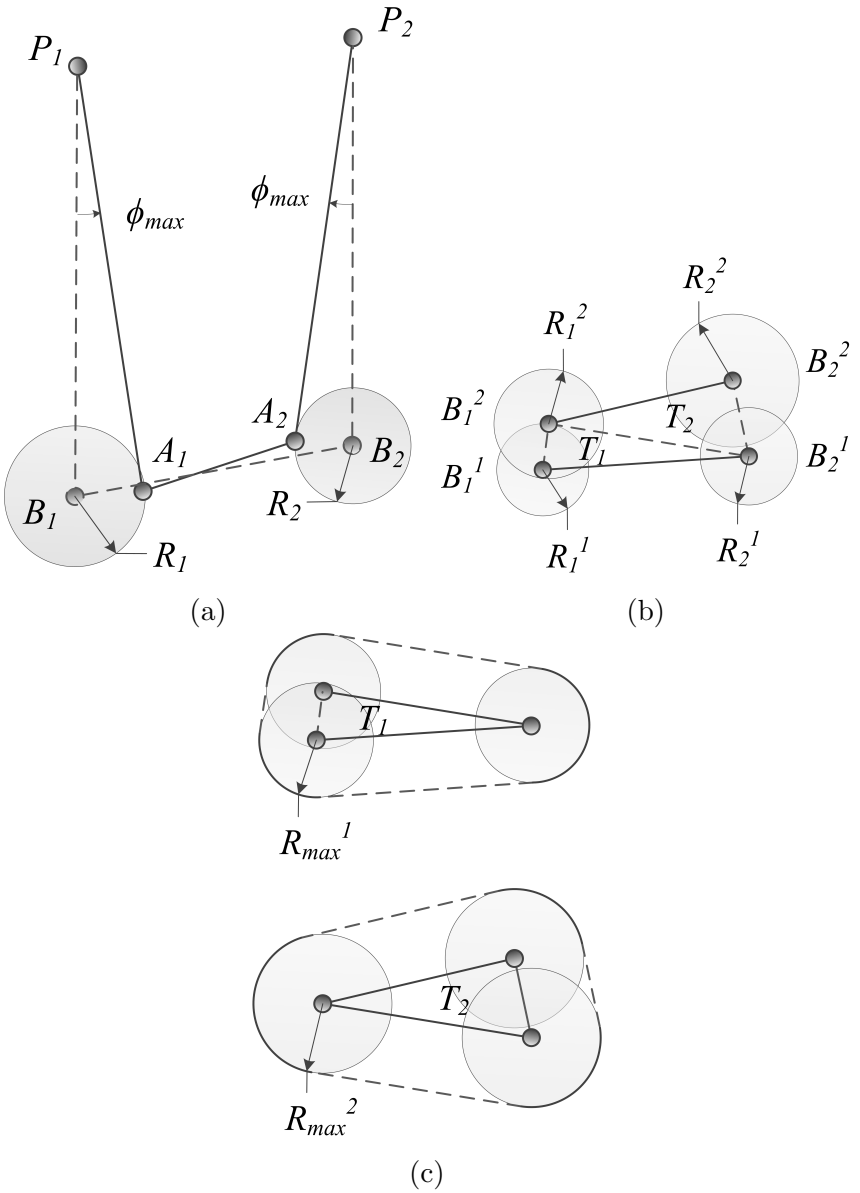


Figure 5.6: Continuous collision detection of the lifting targets: (a) swinging threshold of the attach anchors; (b) triangles approximating the swept path of the lifting target during neighboring steps; and (c) volume generated by the CCD triangles with dilation factor.

Radii R_1 and R_2 of the neighborhoods are written as:

$$R_1 = (L_1 + r)sin(\phi_{max}) \tag{5.46}$$

$$R_2 = (L_2 + r)sin(\phi_{max}) \tag{5.47}$$

which are simplified to:

$$R_1 \approx (L_1 + r)\phi_{max} \tag{5.48}$$

$$R_2 \approx (L_2 + r)\phi_{max} \tag{5.49}$$

since ϕ_{max} is a small angle.

For any two configurations c^1 and c^2, the estimated attach anchors B_1^1, B_2^1 in c^1 and B_1^2, B_2^2 in c^2 form two triangles $B_1^1B_1^2B_2^1$ and $B_1^2B_2^1B_2^2$ (Figure 5.6(b)). The triangle also inherits the error radii R_1^1, R_2^1, R_1^2 and R_2^2. As a result, two TSSs are constructed: T_1 with vertices B_1^1, B_1^2, B_2^1 and radius $R_{T_1} = max(R_1^1, R_2^1, R_1^2)$ and T_1 with vertices B_1^2, B_2^1, B_2^2 and radius $R_{T_2} = max(R_2^1, R_1^2, R_2^2)$ (Figure 5.6(c)).

The continuous collision detection of the lifting target has now been simplified as the collision check between the TSSs and points in the depth map. The collision detection workflow is described in Algorithm 5.4.

The whole collision detection process including the DCD and CCD using the hybrid C-spaces and the CCD for the lifting target using the TSSs is conducted as GPU kernels to evaluating collision violation values in the MSPGA-based path planner.

5.7 Results and Analysis

5.7.1 Simulation results

In order to demonstrate the simulation of dual-crane lifting operations. The proposed balancing solver is used to simulate dual-crane lifting in an industry plant. Two Terex AC700 terrain cranes of capacity 700 tons are employed to erect a cylindrical equipment. In the specific configuration shown in Figure 5.7, the major crane shares of 47.03% the overall load and the assistant crane shares 52.97% of the load. The sling slopes are 0.68 and 0.61 degrees accordingly for the two cranes. The lifting target is tilted by 60.94 degrees.

Algorithm 5.4 Pseudo-code of the continuous collision detection for the lifting target

for all String s^i_{dual} in the population P_{dual} **do**
  for all Edge e_j in string s^i_{dual} **do**
    $B^1_1 \leftarrow Crane1ForwardKinematics(c_j)$;
    $B^2_1 \leftarrow Crane1ForwardKinematics(c_{j+1})$;
    $B^1_2 \leftarrow Crane2ForwardKinematics(c_j)$;
    $B^2_2 \leftarrow Crane2ForwardKinematics(c_{j+1})$;
    $T_1 \leftarrow (B^1_1, B^2_1, B^1_2, max(R^1_1, R^1_2, R^2_1))$;
    $T_2 \leftarrow (B^2_1, B^1_2, B^2_2, max(R^1_2, R^2_1, R^2_2))$;
    $AABB_{T1} \leftarrow Bound_{x-y}(T_1)$, $AABB_{T2} \leftarrow Bound_{x-y}(T_2)$;
    for all point p in the plant depth map which lies within $AABB_{T1}$ **do**
      $d \leftarrow Proximity(p, B^1_1 B^2_1 B^1_2)$;
      if $d <= R_{T1}$ **then**
        $\delta(e_j) \leftarrow 1$; Report collision;
      end if
    end for
    for all point q in the plant depth map which lies within $AABB_{T2}$ **do**
      $d \leftarrow Proximity(q, B^2_1 B^1_2 B^2_2)$;
      if $d <= R_{T2}$ **then**
        $\delta(e_j) \leftarrow 1$; Report collision;
      end if
    end for
    if No collision reported **then**
      $\delta(e_j) \leftarrow 0$;
    end if
  end for
  $n_{edge}(s^i_{dual}) \leftarrow \sum_{i=0}^{L_s-2} \delta(e_j)$;
end for

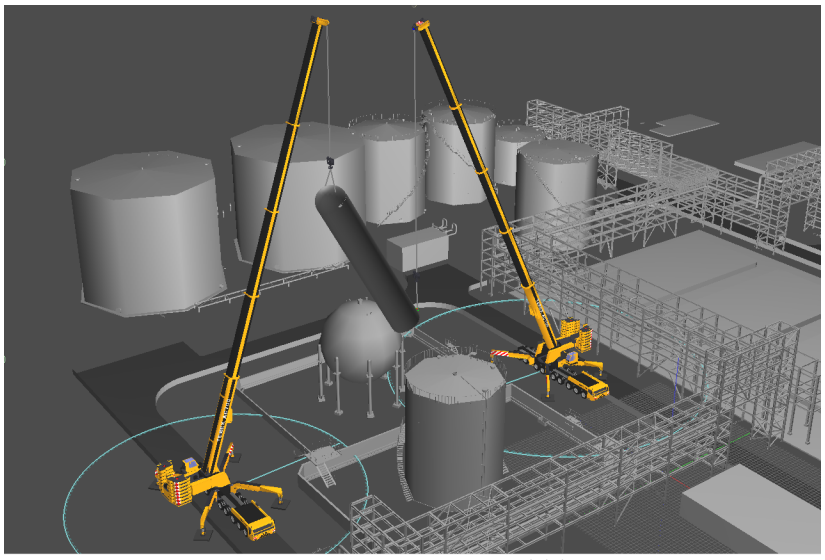


Figure 5.7: Simulation result for dual-crane lifting with sling forces and tilting angles.

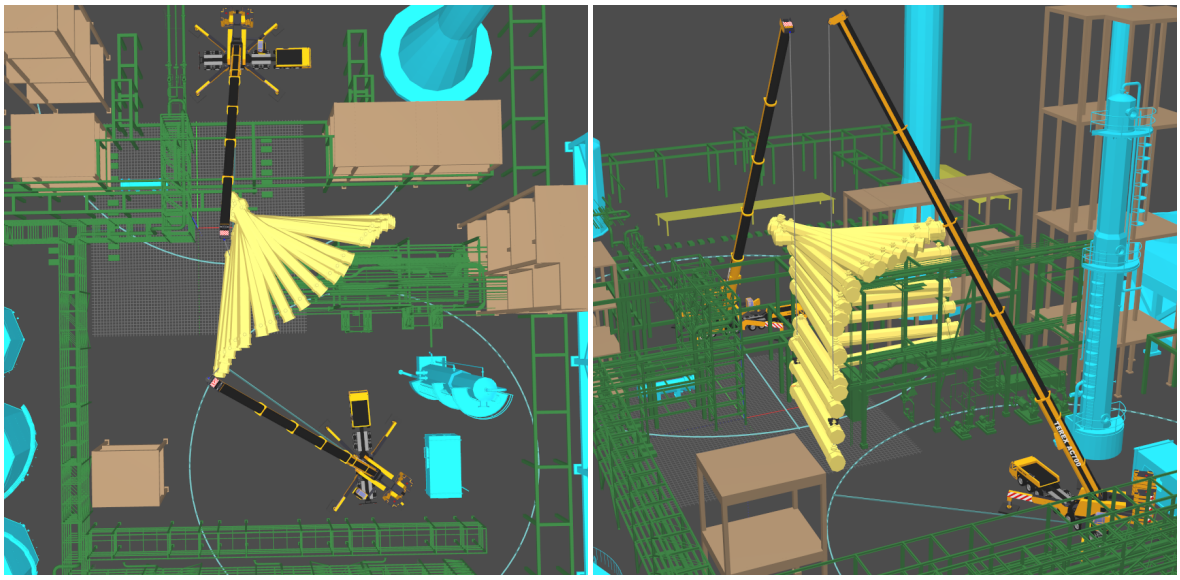
5.7.2 Dual-crane lifting path planning in complex environments

In order to verify that the proposed method supports dual-crane lifting path planning in complex environments, Experiment 5.1 is conducted on a complex industrial plant. The plant model contains 376,205 triangles and 274,108 vertices with a great number of complex piping structures and various equipment. The experiment lifting case moves the 10-meter long target from the ground into the gap between two sets of complex piping structures. The lifting target is required to side-pass the 23 meter high obstacles and rotate the lifting target horizontally for over 90 degrees.

The dual-crane lifting path generated by the proposed method is illustrated in Figure 5.8. The trajectory of the lifting target is displayed in yellow. The lifting path consists of 26 steps. The target is first hoisted to the height of the obstacle and slowly rotated to the destination orientation. Finally, the lifting target is lowered into the gap between the structures. This solution is achieved within 200 iterations which consumes 2s execution time. As shown in the fitness value trend plotted in Figure 5.9, the GA have found a feasible solution near the 50th iteration and come to convergence near the 100th iteration. No violation of collision or coordination happens during the movements. For this specific lifting case, the success rate of the planner is 0.98 out of 1 for 50 trials.

5.7.3 Performance comparison with a GA-based method

To verify the performance of the proposed algorithm, Experiment 5.2 is conducted in the benchmark environment containing three cuboid obstacles proposed in [1]. The lifting target of over 10 meters long is lifted to undergo a parallel movement from one side of the obstacles to another. 200 runs of trials are conducted with both the proposed method and Ali's method. For fairness of the comparison, we also use the proposed parallel collision detection engine in the implementation of Ali's method. Moreover, the original serial components in Ali's method are parallelized by us to benefit from GPU acceleration. Performance of the methods, in terms of success rates and solution qualities with different number of iterations, are shown in Figures 5.10 and 5.11. The success rate of the proposed method in Experiment 5.2 reaches 0.94



(a) (b)

Figure 5.8: Dual-crane lifting path generated in Experiment 5.1: (a) top view; (b) side view

Figure 5.9: Fitness convergence trend in Experiment 5.1

out of 1 in the first 200 iterations while that of Ali's method is still 0.03; With the increase of iteration number, the success rate of the proposed method approaches to 1. With 2000 iterations, the proposed method is able to achieve a success rate of 0.995 out of 1, about 54% higher than that of Ali's method. Figures 5.11(a) and 5.11(b) illustrate the solution qualities measured in both the proposed fitness function and

Ali's fitness function. The trend is similar to that of the success rates. At finishing the first 200 runs, the ratio of the average fitness value in the proposed method with that in Ali's method is nearly 11 when measuring with Ali's fitness function. The ratio is around 29 when both measured with the proposed fitness function. At the end of 2000 iterations, the average fitness value of the best chromosome achieved by the proposed method is 41∼54% higher than that obtained by Ali's method. In this specific lifting case, the average iteration required for finding feasible solutions using the proposed method is 61 iterations (see Table 5.8), far smaller than the 743 iterations required by Ali's method. The time required to complete one iteration using the proposed method is slightly higher than using Ali's method (this is the result with GPU parallelization and the proposed collision detection engine. The original planning time stated in [1] is 12 min). This is because the proposed method considers more constraints such as continuous coordination.

The lack of consideration for internal movements between path steps in Ali's method leads to possible violation of constraints in the output paths. As an example, two dual-crane lifting paths output from the Ali's method and the proposed method are shown in Figure 5.12. The path generated with Ali's method, even though looks smoother and shorter than that in Figures 5.12(c) and (d), violates the coordination constraint in five sections of movements (highlighted in red).

In order to produce a clearer view of the difference. Experiment 5.3 is conducted to compare the sling angles, extra loads caused by sling angles, tilting of the lifting target and the coordination errors $e = |D_{12}^2 - L_{12}^2| / (D_{12} cos(\phi_0))$ between paths produced by the two methods. Ten successful runs of executions are tested for both the proposed method and Ali's method. Figures 5.9 and 5.10 show the maximum and average values respectively. The maximum sling angle of the cranes during conduction of the paths generated with Ali's method are 3 times larger than that for the paths generated with the proposed method. The tilting of slings causes up to maximum 3.28% more load shared on one of the cooperative cranes, which leads to higher overloading risks. This risk may also be measured by the coordination errors, which is 72.8% smaller in the paths generated with the proposed method. On the other hand, the average tilting of the lifting target in the paths generated with the proposed method is maintained

Table 5.8: Comparison on the number of iterations required for finding feasible solution and the execution time for each iteration with Ali's method (Experiment 5.2)

	Number of iterations	**Time per iteration (ms)**
The proposed method	61	8.28
Ali's method (GPU parallelized and with the proposed CD engine)	743	7.64

Table 5.9: Comparison on the balancing properties in the sample paths output by Ali's method and the proposed method in Experiment 5.3 (maximum values)

	Maximum sling angle (degree)	**Maximum extra load (%)**	**Maximum tilt angle of the lifting target (degree)**	**Maximum coordination error (cm)**
The proposed method	1.82	1.26	7.14	266.68
Ali's method	5.38	3.28	18.4779	980.05

Table 5.10: Comparison on the balancing properties in the sample paths output by Ali's method and the proposed method in Experiment 5.3 (average values)

	Average sling angle (degree)	**Average extra load (%)**	**Average tilt angle of the lifting target (degree)**	**Average coordination error (cm)**
The proposed method	1.69	0.84	5.31	240.30
Ali's method	3.56	2.31	14.72	533.19

around 5 degrees, nearly 64% reduction compared with the paths generated with Ali's method.

The above results show that the proposed method has largely improved the convergence of the GA searches and is able to achieve higher quality solutions within much shorter time. The proposed method is able to output well-coordinated paths for the whole conduction process.

5.7.4 Comparison with a PRM-based method

The effectiveness of the proposed algorithm is also compared with a previous one [2] based on the state-of-art PRM path planning method which samples configurations in the C-space to construct road maps. Their method have simplified the problem by constraining both the sling angles and the tilting angle of the lifting target so that the dual-crane lifting system can be solved by inverse kinematics. An optimal path generated in the proposed algorithm is shown in Figure 5.13 together with two paths presented in Chang's paper [2]. Both paths from [2] let the lifting target follow linear motions interpolating key-frame locations in the Euclidean space, one of which climbed along the obstacles. The other path traveled around the main obstacle. The underlying problem of paths is brought by the fact that the tips of crane booms who are driving the lifting target through cables are only allowed to move on a sphere surface. Linear motions of the end effector, when projected back to the polar system of the configuration space, would be non-linear arcs. Comparatively, the path generated by the proposed algorithm appears to be slightly irregular in the Euclidean space. But, when the path is drawn in the C-space, it reveals a linear, smooth and axis-aligned pattern which is much easier for the cranes and human operators to track. On the other hand, Chang's algorithm relied on zero tilting angle of the lifting target. This constraint reduced the complexity of the planning by reducing two degree of freedom for the PRM searches. However, This assumption will be invalid when the cranes are asked to fulfill erection tasks requiring tilting of the lifting target which are frequently demanded in industrial applications. Consequently, the proposed algorithm performing forward kinematic planning using GA is able to produce optimized paths in a more smooth and nature way. By accepting tolerable sling angles and allowing tilting of the lifting target, it also suits for more types of demands.

5.8 Summary

A method for simulating and automatic planning dual-crane lifting operations is proposed in this chapter. The dual-crane lifting kinematics is decomposed into the manipulation sub-system and the suspension sub-system. The suspension system is modeled

through forming and solving a nonlinear equilibrium equation system. Dual-crane lifting path planning is formulated as a nonlinear optimization problem with collision constraints, coordination constraints, and DOF constraints. The optimization problem is solved by a GPU-based and LGP-enhanced MSPGA to obtain optimized lifting paths. A TSS based method is introduced to deal with the CCD of the lifting target.

The method is able to provide optimal dual-crane lifting paths which are smooth, collision-free and properly coordinated. The output paths by algorithm maintain the balance of the lifting target properly. Due to the increased complexities of the dual-crane lifting problem, the dual-crane planner takes more iterations to enter the feasible region compared to the single-crane planner. But it still shows fast convergence even in complex and narrow feasible regions.

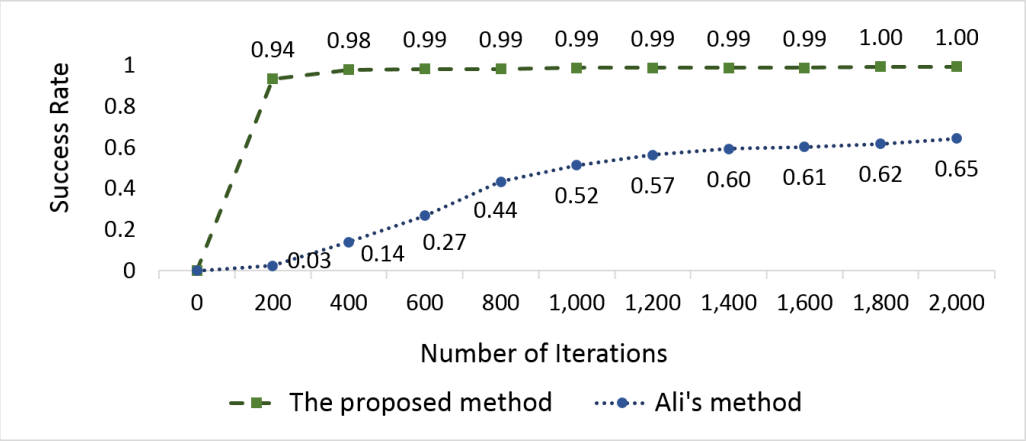
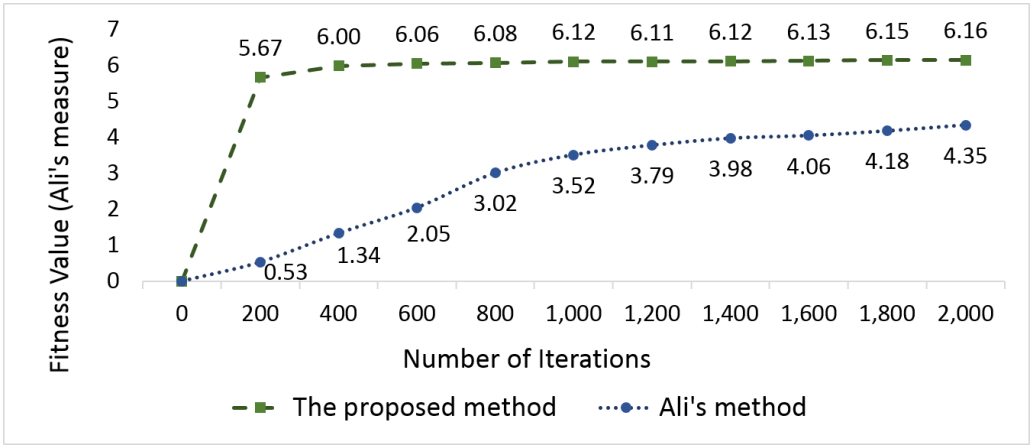
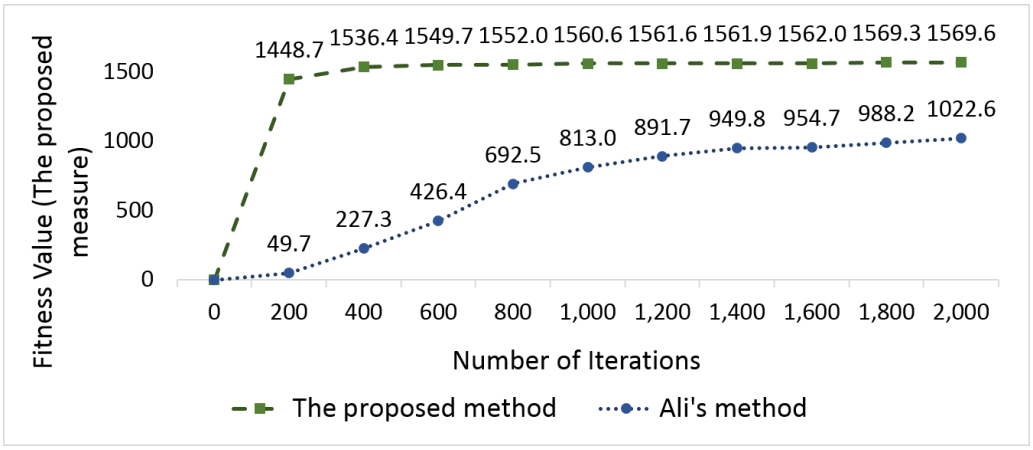


Figure 5.10: Comparison of the success rate using Ali's method and the proposed method under different numbers of iterations (Experiment 5.2)

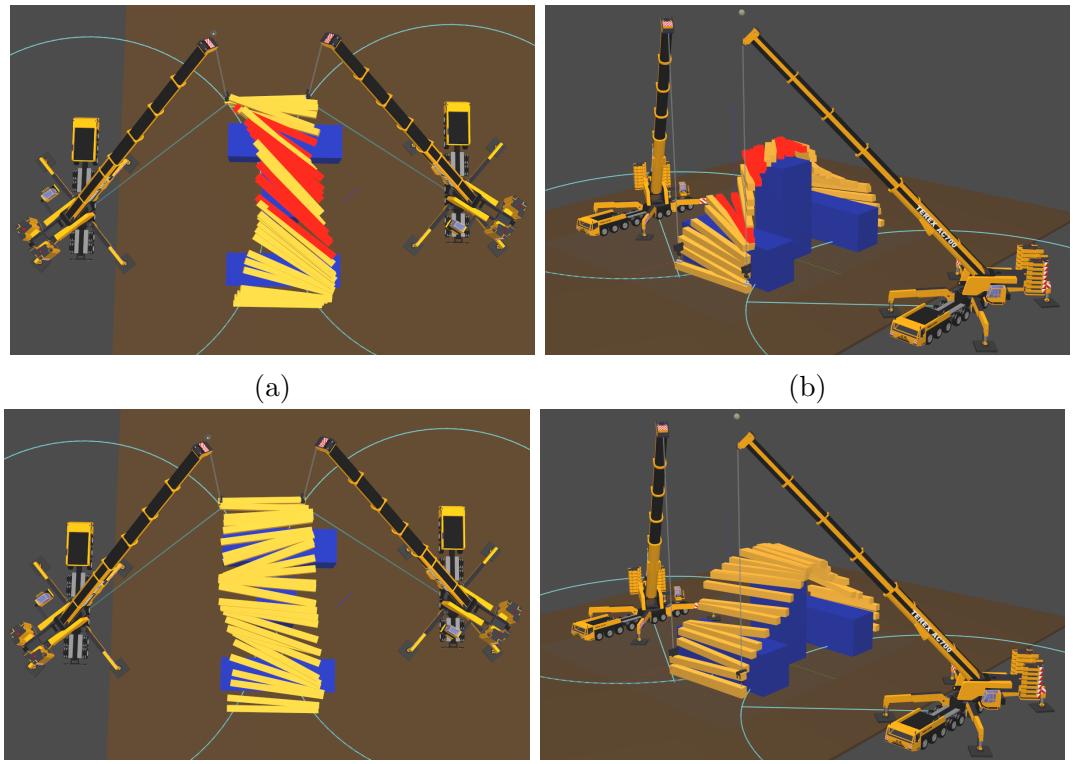


(a)



(b)

Figure 5.11: Comparison of the solution qualities using Ali's method and the proposed method under different numbers of iterations (Experiment 5.2): (a) using Ali's measure; (b) using the proposed measure.



(a) (b)

(c) (d)

Figure 5.12: Sample path generated with the method of [1]: (a) top view; (b) side view; And the path generated with the proposed method: (c) top view; (d) side view.

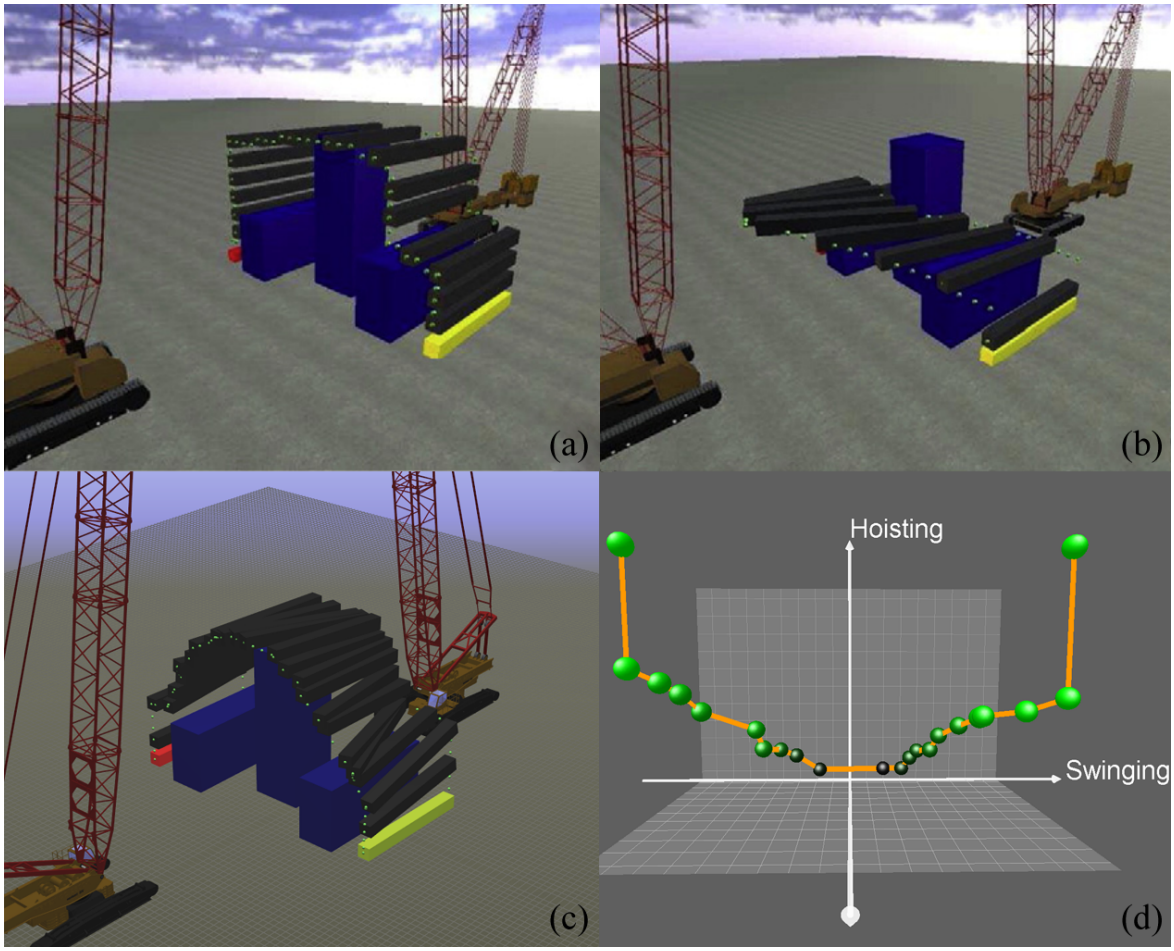


Figure 5.13: Comparison of two paths generated by: (a) and (b) the method of [2] and (c) the proposed method. (d) is the C-space path of the major crane conducting the lifting path shown in (c). Lighter green stands for smaller luffing angle.

Chapter 6

System Design of the Lift Planner cum Crane Simulator

6.1 Introduction

Crane accidents may happen due to various reasons. Inadequate planning, lack of knowledge, bad communication, as well as machine and human errors may all lead to severe accidents causing unpredictable financial loss, equipment damage, time delay and even death [149, 150]. CALP systems aim to improve the safety and productivity in lifting operations. Systems for interactive lift planning enable users to define lifting paths with trial and error approaches in a simulation environment. However, this approach can not produce high-quality solutions efficiently especially for novice users. Therefore, automatic and optimal lift planning with minimum user interference is increasingly desired in the industry. This chapter aims to design a practical and efficient lift planner cum crane simulator which can fulfill the requirements of automatic lift planning and vocational training.

Prior efforts on CALP systems have focused on the use of simulations to assist interactive lift planning. These systems have been designed to assist mechanical checking, safety monitoring, and evaluations for interactive lifting paths. HeLPS is a conceptual

This paper is based on the book chapter: P. Cai, Y. Cai, I. Chandrasekaran, and J. Zheng, "A GPU-enabled parallel genetic algorithm for path planning of robotic operators", in Y. Cai and Simon See (eds.), GPU Computing and Applications, pp. 1-13, Springer, 2015 and the conference paper: P. Cai, C. Indhumathi, and Y. Cai, "High-performance simulation for vocational training of heavy mobile cranes", the Second Asia-Europe Symposium on Simulation and Serious Games, October 2014, Zwolle, the Netherlands.

design of CALP systems introduced by Hornaday *et al.* in [121] based on interviews and lift studies with experts. Lin and Haas [122] have developed a COPE (short for Critical Operations Planning Environment) system being able to perform the site analysis, equipment placement and lifting path generations on 2D drawings. Varghese *et al.* [123] have implemented part of the functionalities described in the HeLPS system design. Their system enables interactive lifting by simulating crane operations in a 3D environment and monitoring safety factors during interactions. Chadalavada and Varghese [124] have developed a CLPS simulation system as a plug-in approach of the Autodesk Inventor. Their system is able to assist common lift planning components such as crane selection and setup, pick & place location determination and interactive lifting operation planning.

Although the 2D drawings used in some systems mentioned above are easily accessible in the lifting industry, most of them are more or less simplified the information in the vertical direction and need to be combined with many non-graphical data to feature the environment and lifting tasks. Other systems using conventional CAD models can simulate more realistic environments. However, building the CAD models from the scratch for complex environments (especially for those existing old plants) would be highly time-consuming with no guarantee for the desired accuracy of the resulting models. Therefore, previous systems mentioned above are usually more suitable for training and interactive lifting planning.

Unlike the existing solutions, the proposed lift planner cum crane simulator can deal with actual industrial environments accurately to perform automatic lift planning. The system makes use of both data in mesh and point cloud forms. Many new industrial plants have digital versions in the PDMS, Smartplant systems or BIM systems. In this case, the system developed extracts geometric information from these digital plants. For old plants or sites that do not have digital versions, the proposed system uses laser scanning to capture high-resolution point cloud data for representing the exact geometric information. Based on these accurate digital environments, the proposed system enables automatic lift planning by implementing and integrating the collision detection algorithm and the lifting path planners for single and dual cranes introduced in previous chapters.

This chapter firstly presents the overall design of the system followed by a detailed introduction on its functional components. In these contents, the processing procedures for handling PDMS and point cloud data are also discussed. GPU implementations of the algorithms introduced in previous chapters are elaborated to show how the hierarchical architecture of CUDA can be fully utilized. Finally, a case study is provided to illustrate the potential applications of the proposed system for real-life industrial projects.

6.2 System Architecture

Figure 6.1 illustrates the overall system structure of the lift planner cum crane simulator which consists of four engines. The digital cranes considered by the system are mobile cranes whose structures are defined in Section 4.2. The planner cum simulator acquires data from PDMS, BIM and laser scanning technologies to represent complex industrial environments. These geometric data together with the digital crane from the database are imported to the modeling engine and organized into a scene graph. The interaction engine receives user inputs from the GUI and interactive devices to manipulate the cranes or launch lifting path planning tasks. This engine handles the interactions between cranes, the lifting target, as well as the environment, and produces safety warnings for invalid operations. The safety monitoring is achieved by incorporating the collision detection algorithm introduced in Chapter 3 and the balance solver for the dual-crane suspension sub-system introduced in Chapter 5. The optimization engine takes the user-defined lifting task specifications to produce automatic optimal path suggestions. This is realized by implementing the MSPGA-based path planners introduced in 4 and 5. The optimization engine can also be used to evaluate of user-defined lifting paths. Finally, the visualization engine collects information on the scene and lifting paths to produce 3D visual outputs and to display lifting animations.

6.2.1 Hardware and software components

The hardware components supporting the system is a normal PC equipped with 3.40 GHz Intel Core(TM) I7-3770 processor and 16GB system memory based on windows

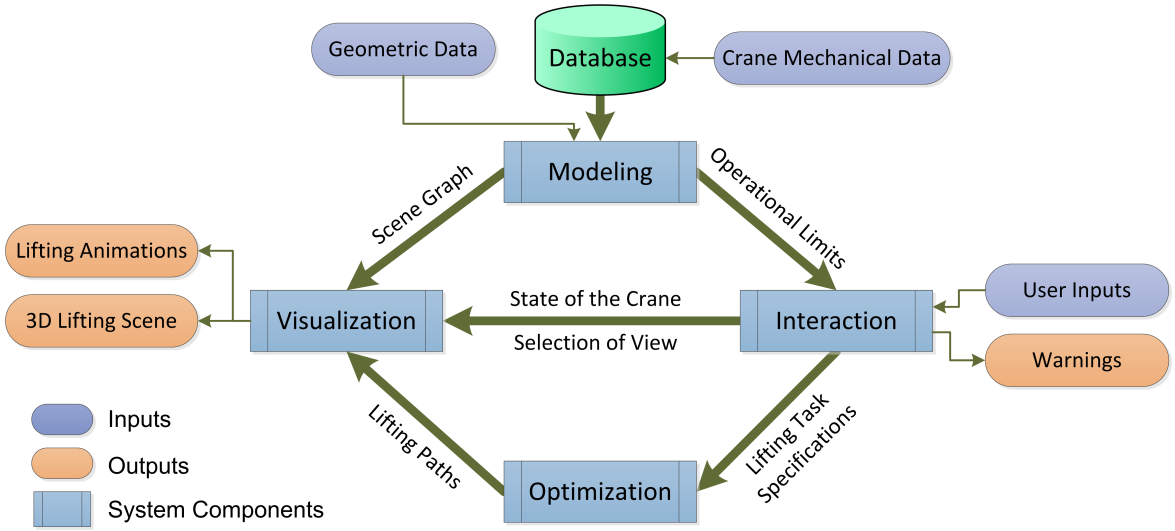


Figure 6.1: System architecture of the lift planner cum crane simulator.

operating system. The basic components of the lift planner cum crane simulator such as modeling, visualization, and crane manipulation are handled by the CPU. More computational intensive components such as real-time collision detection and automatic path planning are handled by the GPU. The use of GPU in collision detection and path planning can significantly improve the performance of the system. This system uses the NVIDIA Tesla K20c graphic card which possesses 5GB graphic memory and 2496 CUDA cores.

The proposed lift planner cum crane simulator is developed using C++ and is compatible with both Windows XP and Windows 7 platforms. OpenGL is used to develop the rendering engine performing transformation, rasterization (for rendering) and display of models. Microsoft Foundation Class (MFC) is used for designing the Graphical User Interface (GUI) of the system. GPU-based collision detection (presented in Chapter 3) and path planning algorithms (presented in Chapters 4 and 5) are developed using CUDA C/C++. CUDA has a hierarchical architecture which reflects the hardware structure of NVIDIA GPUs. The parallel algorithms in this system are optimized to take advantage of the hierarchical CUDA architecture. Details of the implementations will be introduced in Section 6.2.2.3.

6.2.2 Engines and supporting methodologies

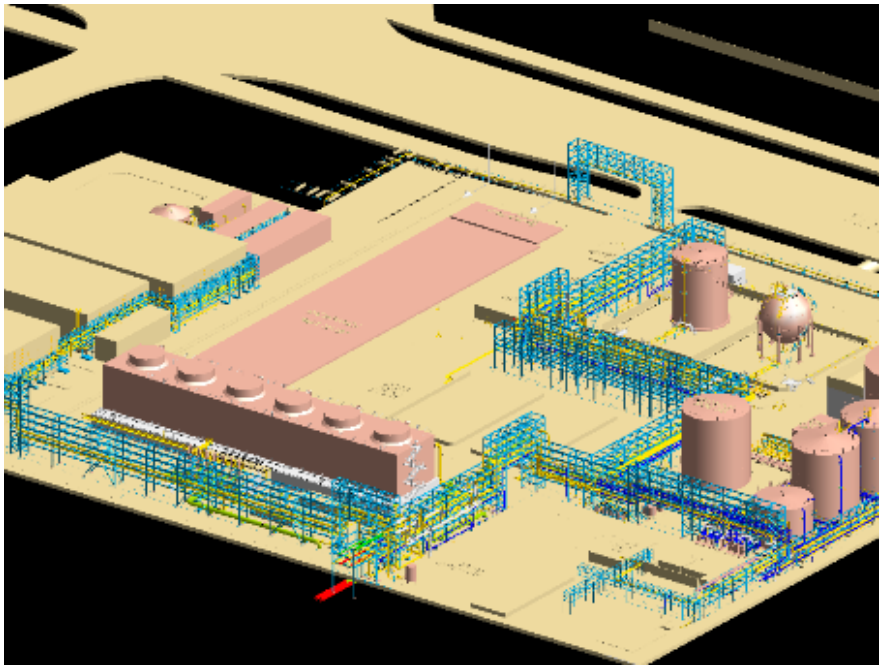
This section provides a detailed presentation of the functional engines and their supporting technologies, especially on how data from PDMS, BIM and laser scannings are processed and made use of by the system to represent complex industrial plants and sites. These technologies usually produce huge data sets which are difficult to be used efficiently in simulation environments. In the proposed system, all these data are unified into the MDM representation for simulation, interactions and optimization. In this way, the proposed system deals with meshes and point cloud models uniformly for collision detection and path planning.

6.2.2.1 Modeling engine

The modeling engine of the planner cum simulator system takes care of both the digital cranes and the digital environments. For the digital cranes, the engine manages their geometries, kinematic structures and mechanical data. For the environments, the engine processes different types of raw data into accessible formats and constructs a unified representation for them. As a result, the modeling engine produces a scene graph which organizes all the above-mentioned data in a hierarchical way.

The proposed modeling engine maintains a database of digital cranes which records all their geometrical, mechanical and operational properties. The CAD models of digital cranes are constructed using the actual structures and measures of the physical cranes. When a digital crane is imported into the system, a branch of the scene graph is constructed following the actual kinematic structures of the cranes. Therefore, different types of cranes result in different branch structures in the scene graph. This difference in kinematics also affects how the hybrid C-spaces, ASVs, and TSSs are constructed in the lifting path planners.

In the scene graph branches of cranes, internal nodes record transformations to be applied on their child nodes, and components of cranes are linked to leaf nodes. OBBs are also constructed for these crane components for later use in collision detection and path planning. The lifting targets are initially outside the crane scene graph. They are inserted into the crane scene graph when attached by one or more cranes.



(a)

(b)

Figure 6.2: Digital environments used in the proposed lift planner cum crane simulator system: (a) A PDMS plant; (b) A point cloud plant.

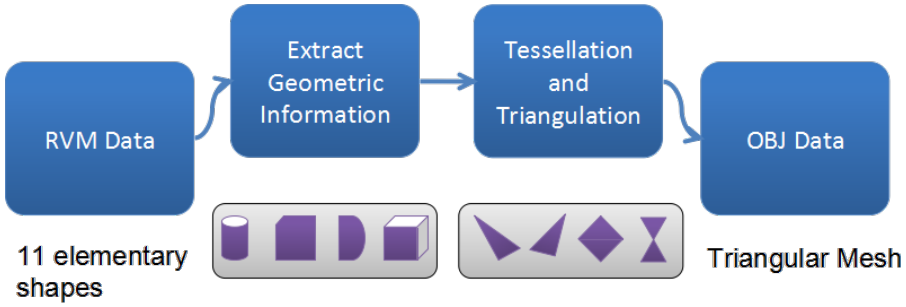


Figure 6.3: Pipeline for processing PDMS data

An example of a digital plant from the PDMS system is shown in Figure 6.2(a). In order to make use of the geometric data in the PDMS system, the proposed system first converts these data into triangular meshes. Figure 6.3 shows the processing pipeline for the PDMS data. Initially, all objects in the PDMS system are represented as eleven elementary parametric shapes. The proposed system extracts these parametric data such as positions, transformations, radii and height values, and uses a tessellation to convert them into polygon meshes. Afterwards, the triangulation process takes over the polygons and decomposes them into triangles. These triangles, represented by vertices, normal values, faces, and texture coordinates are written into the hard disk for further use. Similar pipelines are applied when dealing with data from Smartplant and BIM systems.

To deal with old plants who do not have digital versions and construction sites changing after each day's work, this research uses laser scanners to capture the environment in 3D point cloud form (Figure 6.2(b)). High accuracy point clouds can keep the fine details for complex industrial environments. Therefore, the proposed system makes use of the point clouds for both simulations and path planning.

However, the raw data produced by laser scanning are not directly usable. They have to go through a processing procedure to produce a usable point cloud models. 6.4 illustrates the processing procedure for point cloud data. Capturing the whole industrial environment involved in a lifting task often requires multiple laser scans. These point clouds produced by these scans are initially located in their own coordinate systems. Therefore, a registering (stitching) process is required to unify the coordinate frames of multiple scans. This stitching process is achieved by matching feature points

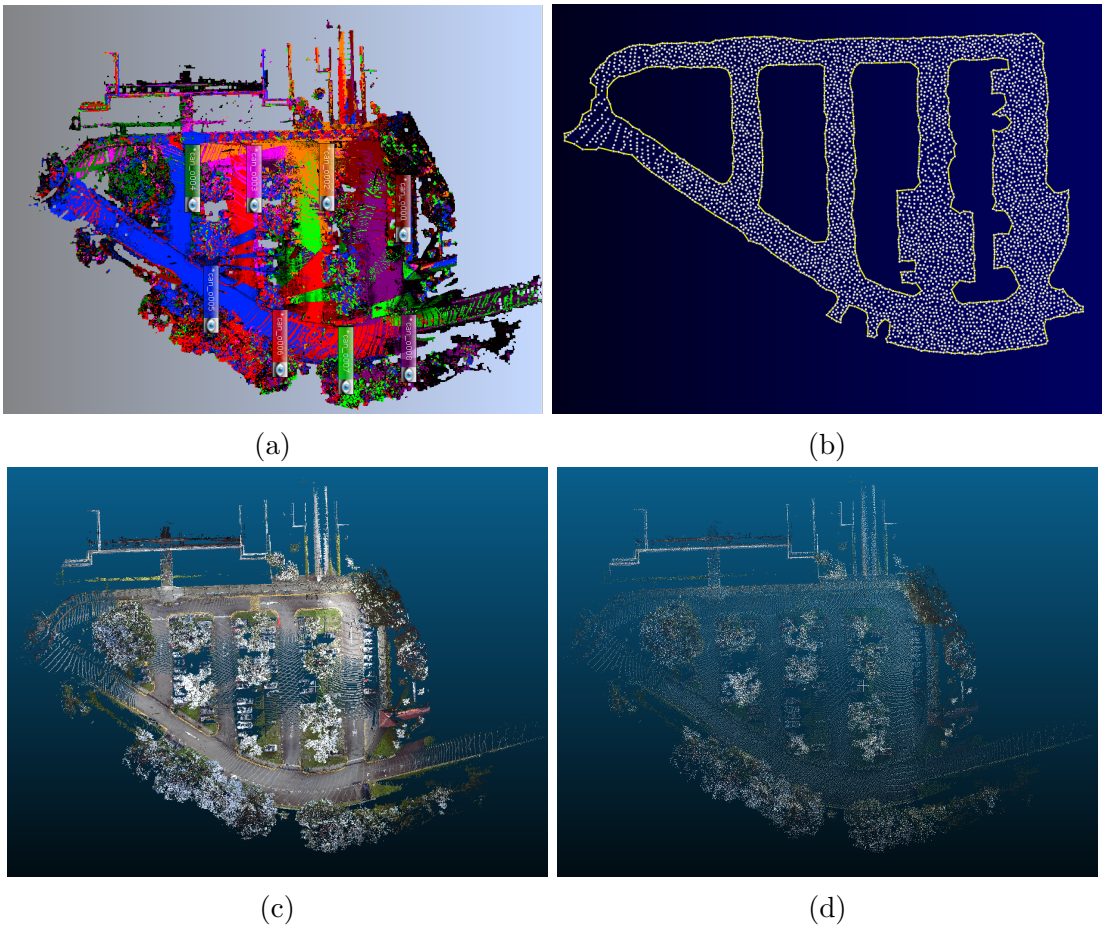


Figure 6.4: Steps for processing point cloud data: (a) registration of multiple scans; (b) feature extraction; (c) colored data; (d) sub-sampling.

in different scans. The resulting data still possess all original points from the registered scans and thus can be very inefficient to use. Therefore, the system also runs a sub-sampling process [151] to down-sample or re-sample the points. Through this process, the size of the final data set is reduced to a tractable scale. In some occasions, the system needs extra geometric information other than points. For instance, when performing crane setup planning, it requires information about the ground. In these cases, the system also runs a feature recognition process [152] to extract the ground points.

When the processing of raw geometric data has finished, the system loads the final data sets and constructs MDMs for them (see details in Chapter 3). The orginal triangular meshes and point clouds are also inserted into the scene graph as an environment

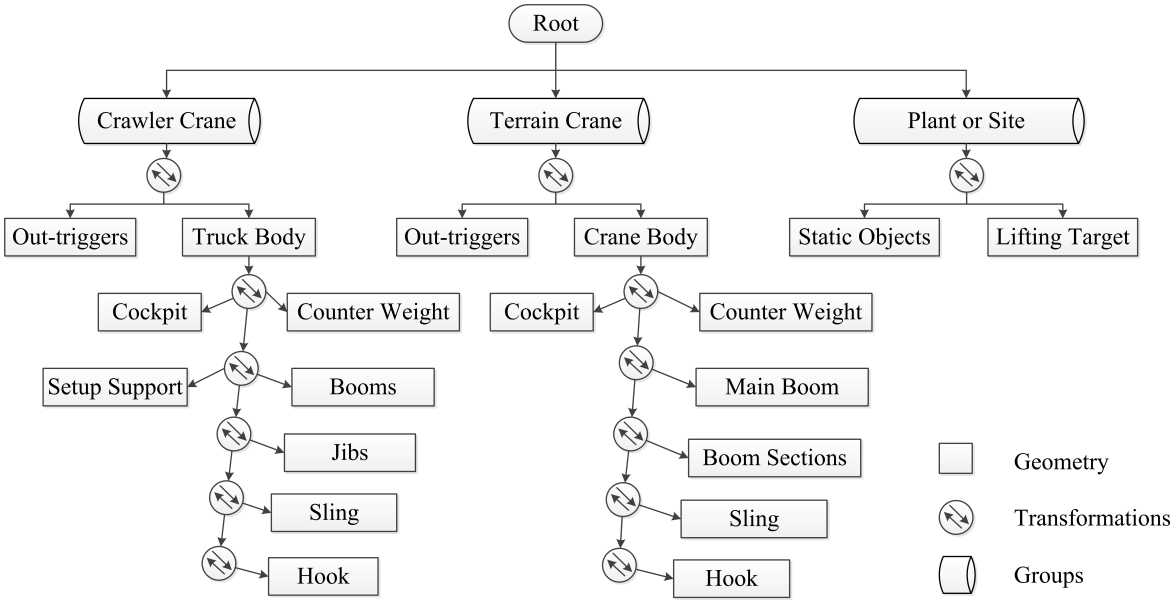


Figure 6.5: Illustration of the scene graph containing the crane models and the industrial environment

branch. Figure 6.5 shows the resulting hierarchical tree structure of the scene graph which usually contains one plant branch and several crane branches.

6.2.2.2 Interaction engine

The interaction engine simulates the crane manipulations for users and evaluates the safety factors during lifting. This engine supports the following four types of interactions:

(i) Crane selection (by using mechanical data in the crane database)

(ii) Crane setup (locating, out-trigger extension, counter-weight loading and so on.)

(iii) Crane manipulation (rotating the cockpit, main boom, jibs and the lifting target, extending booms and the sling)

(iv) Execution of lifting path planning and modification of result paths.

This engine allows users to interact with the scene with keyboards, mouses, and joysticks. When using joysticks to manipulation the cranes, the proposed system

simulates real crane operations. During the simulations, the engine provides proximity warnings and collision responses by implementing the collision detection algorithm introduced in Chapter 3. For dual-crane lifting, it also calculates the tilting angle of the slings and the lifting target, as well as the load shared by the two cranes using the balance solver introduced in Section 5.4.1. In this sense, the system could perform as a training tool to help trainees practice safe lifting operations in realistic virtual environments.

The interaction engine also enables users to define lifting task specifications for automatic planning. The users can also select a most suitable plan from several candidate optimal paths and modify them to meet further requirements. In this case, the system serves as an automatic lift planning tool with interactive improvements enabled.

6.2.2.3 Optimization engine

The optimization engine implements the single-crane lifting path planner introduced in Chapter 4 and dual-crane path planner introduced in Chapter 5 to produce optimal lifting paths for guiding lifting operations. In order to obtain an efficient GPU-enabled automatic lift planning system, the implementations of these parallel path planning algorithms are optimized for the CUDA architecture.

This section presents the GPU implementation of the MSPGA-based path planner for single-crane lifting introduced in Chapter 4 as an example. A summary of the workflow of the planner is first presented. Then the details on the GPU implementations of the algorithms is discussed.

Figure 6.6 provides an overview of the single-crane planner. The planner first generates an initial population in the GPU and passes it to into the evolutionary iterations. In each iteration, the population goes through a fitness evaluation process, a roulette wheel selection, crossover operators, and mutation operators successively. While the flow is controlled by the CPU, GPU handles the functional components of the MSPGA-based path planner following the data flow shown in Figure 6.7.

In each iteration, a fitness evaluation kernel is first launched to generate the fitness value pool for the population in GPU memory. The kernel parameters of the fitness

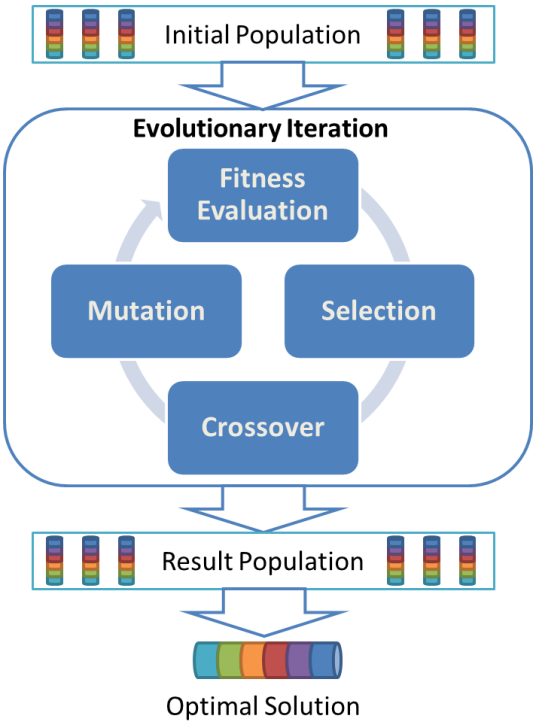


Figure 6.6: Overall flow of the MSPGA-based path planner.

evaluation kernel are defined as $<<< N, L, 1 >>>$ where N is the population size and L is the chromosome length. The GPU implementation is shown in Figure 6.8. Each chromosome in the population is assigned to a thread block while genes in the string are handled by separate threads. Within each thread, continuous collision detection is done for genes. A synchronized shared memory unit is used to count the number of collision violations (denoted as n_i in the Equations 4.21 and 5.41) and coordination violations (denoted as m_i in the Equations 4.21 and 5.41) in the chromosome. These counted numbers are further used to calculate the fitness values. The kernel returns when all fitness values are settled in the GPU global memory.

These fitness values are then normalized and accumulated to produce a chance pool encoding the chance of survival of each chromosome. The mating pool is represented as indexes of chromosomes instead of actual chromosomes. In order to achieve global communications in parallel processors, a specially designed parallel kernel is used to generate the mating pool (as shown in Figure 6.9). Firstly a CUDA kernel is launched with kernel parameters specified as $<<< 1, N, 1 >>>$ where N is the population size. Then, a uniform random float number $r \in [0, 1)$ is generated by each thread and is

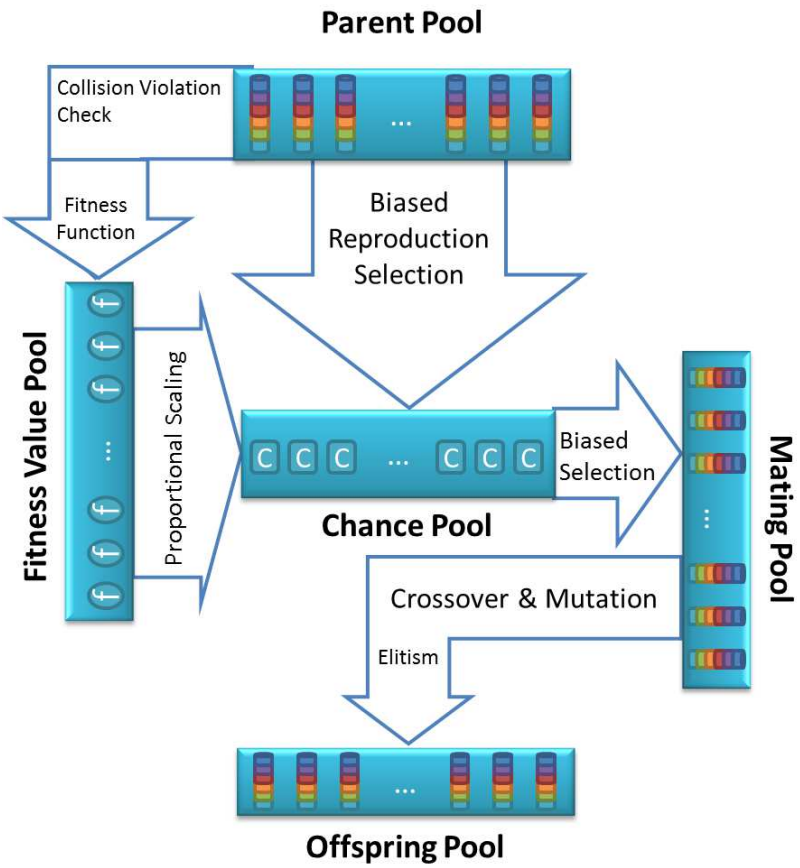


Figure 6.7: Indication of the data flow inside the evolutionary iteration.

compared between the array of survival chances of chromosomes. Suppose that the accumulated chance pool is denoted by $(c_0, c_1, \ldots, c_{N-1})$, then the task of threads is to find the index i that satisfies $c_{i-1} \leq r < c_i$. Here c_{-1} is counted as zero. The index found by this process is then stored in the corresponding position in the mating pool.

A roulette wheel selection is conducted in parallel using information from the chance pool. In this selection process, chromosomes with higher fitness values have better chances to be selected into the mating pool for reproduction. After the mating pool data are settled, the crossover operator is applied immediately (Figure 6.10). The CUDA kernel parameters are specified as $<<< N, L, 1 >>>$. In the GPU kernel, the jth thread block B_j $(j \leq 0)$ handles a pair of chromosomes $(s_{i_{j-1}}, s_{i_j})$ where i_j stands for the index in the jth entry of the mating pool (B_0 is reserved for elitism). Data of the two chromosomes are poured into the shared memory of the block for future use.

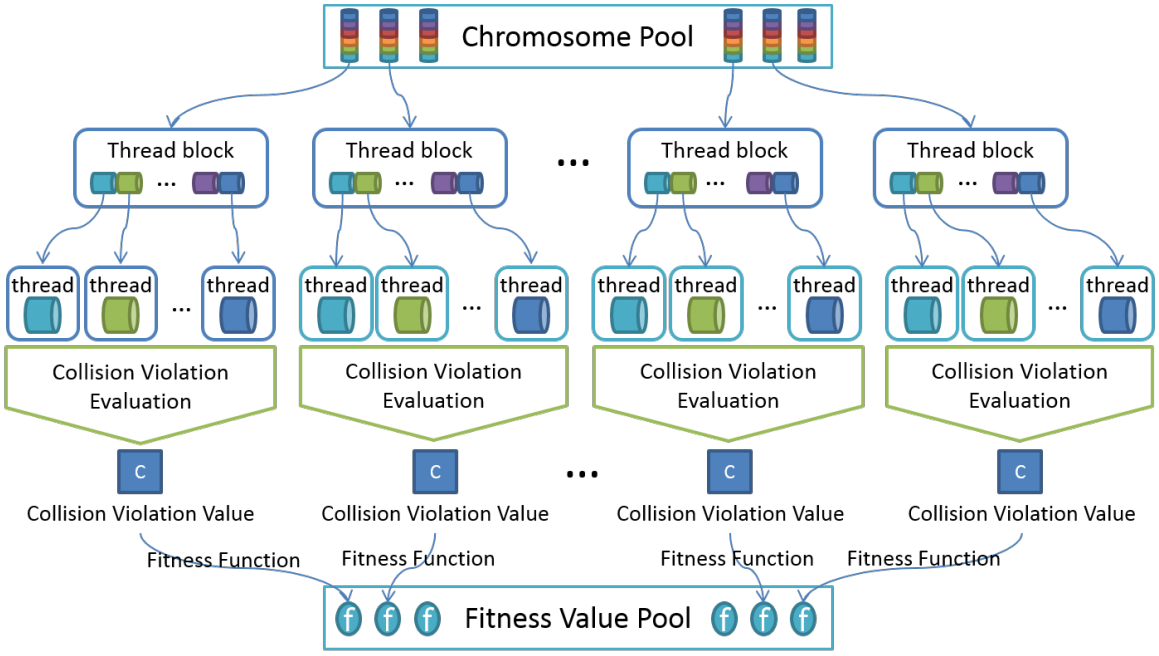


Figure 6.8: GPU implementation details of the fitness evaluation process.

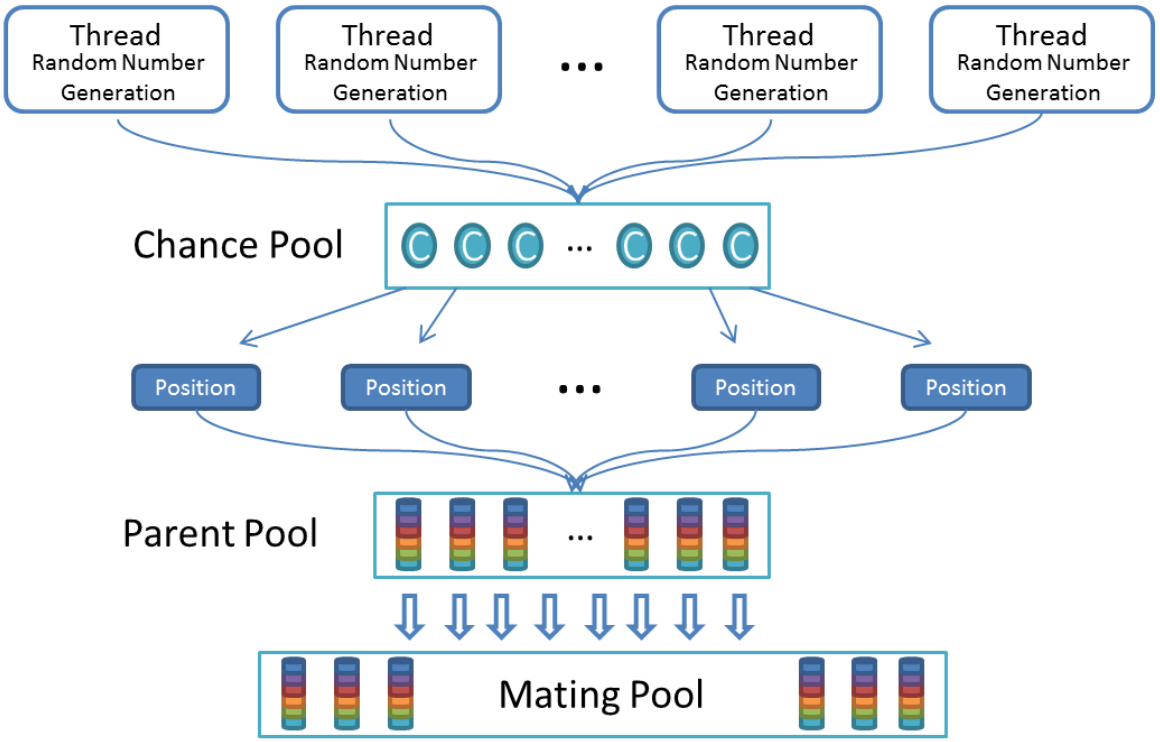


Figure 6.9: GPU implementation details of mating pool generation.

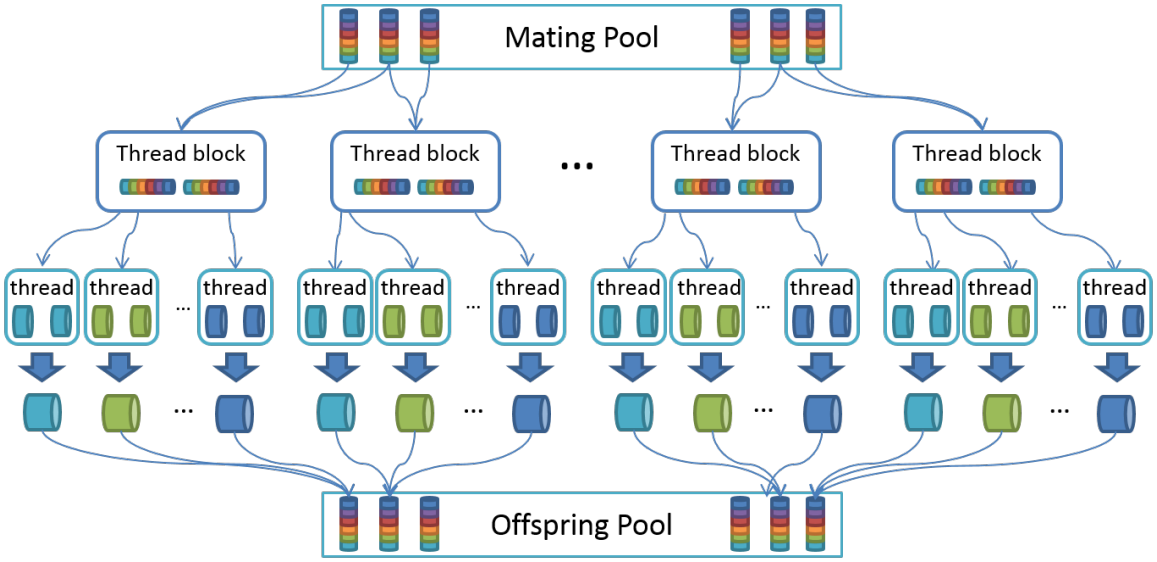


Figure 6.10: GPU implementation details of the parameter-based crossover operator.

The pairs of genes from parents are then assigned accordingly into the parallel threads in the block. The kth thread in the jth block retrieves g_k^{j-1}, g_k^j and their neighboring genes into its local c. The two parent genes are compared by applying the crossover strategies presented in Table 4.6 for single-crane cases and the strategy stated in Algorithm 5.1 for dual-crane cases. After all gene positions are determined, the kernel terminates and the offsprings remain in the GPU global memory.

Finally, adaptive mutations using the fitness values are conducted. The mutation tasks for the population are distributed into the GPU blocks and threads in a similar way. After the mutation, the MSPGA enters the next iteration or outputs results if the termination criteria have been met.

6.2.2.4 Visualization engine

The visualization engine takes the scene graph from the modeling engine and renders the scene using OpenGL. It also visualizes the outputs from the interaction engine and the optimization engine. For example, when performing automatic lift planning, this engine animates the optimal lifting paths output by the planners. Users can thus examine the paths and communicate the plan with team members or clients. The lifting animations can be displayed with stereoscopic rendering to enhance the

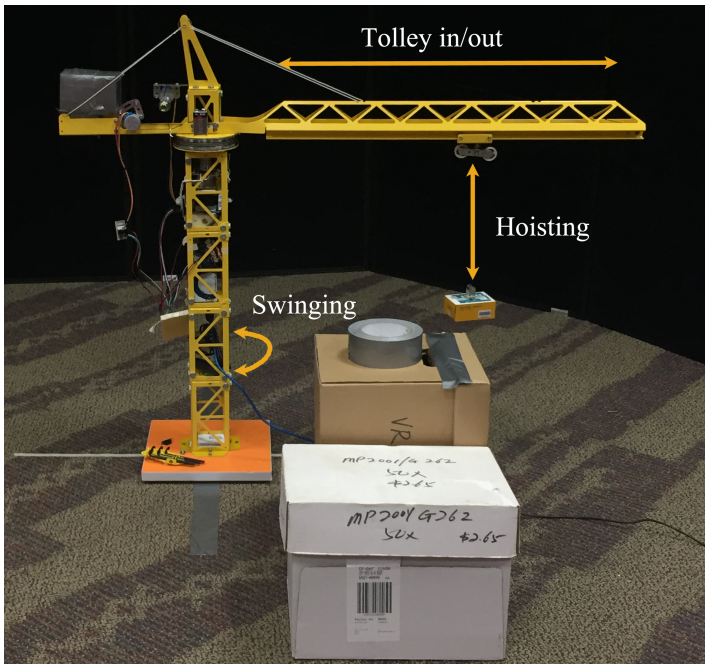
visualization of the paths. For simulation purpose, the engine displays safety warnings in occasions when the cranes are proximate to the obstacles or when the balancing of the lifting target has failed in dual-crane lifting.

6.3 Results and Discussions

The proposed system might be used to guide the automatic lifting for physical cranes. This application can be fulfilled under circumstances where the crane supports automatic control. In other words, the lifting operations can be highly automated. Human interference is only required for defining lifting tasks once the crane is set up on the site. However, automatic control of cranes is not yet favorable due to limitations of hardware developments and government safety regulations as the modification of actual crane for site operation is under very strict control. In this case, the proposed system can perform off-line planning for human operators, signal men and riggers. Lifting paths are generated in the simulation environments based on accurate geometric information for the plants or sites. These data can be acquired from laser scanning or the PDMS system. The lifting path is then used to guide the operation conducted by the lifting team like a GPS car navigation system. Another application of the proposed system is to perform vocational training for crane operators. Unlike training using real cranes which are expensive and time-consuming, trainees can practice safe crane operations in the realistic simulation environments in a cost-effective way.

6.3.1 Validation of automatic lifting with a scaled model crane

To validate the potential and efficiency of the system in guiding automatic lifting, Experiment 6.1 is conducted to test the four steps in automatic lifting: data acquisition, data processing, lifting path generation and lifting path conduction. The experiment uses a scaled physical model of a tower crane which resides in an environment consisting of several obstacles and many other objects in a room (Figure 6.11(a)). Actuators and sensors are added onto the model crane to enable the three basic DOFs of the tower crane: swinging, trolley in/out and hoisting. The model crane communicates with the system using file exchanging. In the software side, a digital version of the crane is modeled and stored in the database.



(a) The model crane and the real environment

(b) The virtual environment in point cloud form

Figure 6.11: The model crane and the environment used in 6.1

Experiment 6.1 is conducted in an environment mainly containing two box-shaped obstacles and a cylinder-shaped obstacle. The environment is captured by laser scanning to produce a high-definition point cloud which combines 4 scans taken at different angles. The final point cloud shown in Figure 6.11(b) contains 21,000,716 points with a resolution less than 1 millimeter. Even using this large point cloud, the simulation

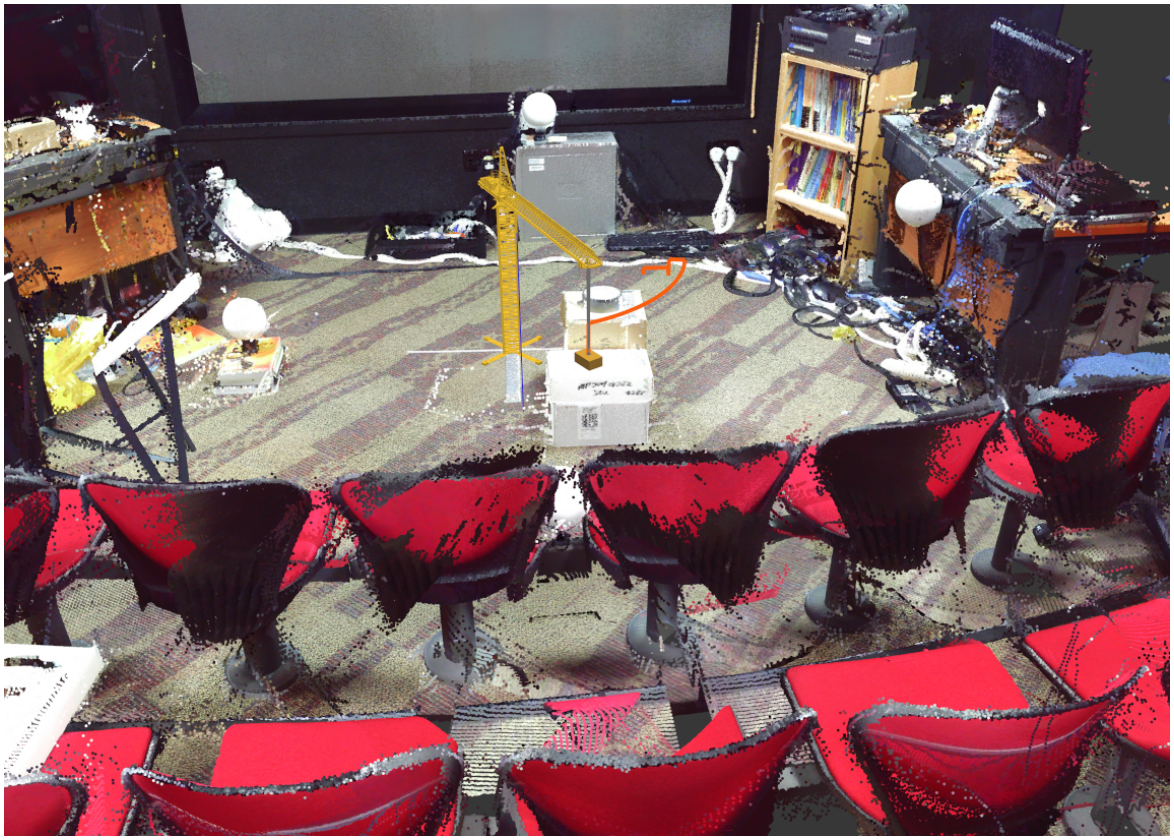


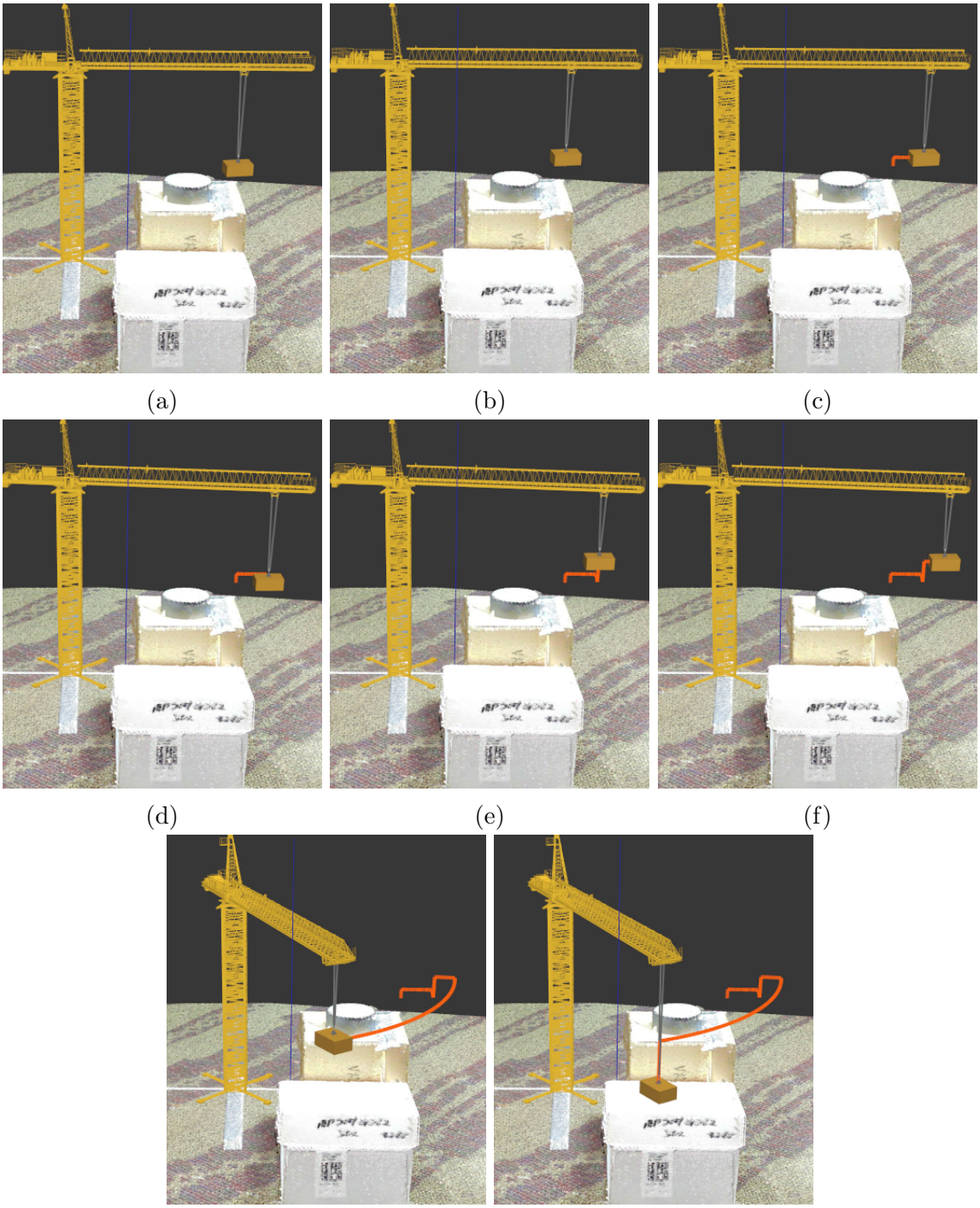
Figure 6.12: Lifting path used in Experiment 6.1 displayed in the point cloud environment

Table 6.1: Node configurations in the lifting path generated in Experiment 6.1

Step	**Operation Type**	**Amount**
1	Hoisting	2.68 cm
2	Trolley out	8.88 cm
3	Swinging	-7 degrees
4	Hoisting	6.27 cm
5	Trolley out	5.65 cm
6	Swinging	-58 degrees
7	Hoisting	-12.93 cm

performs in real-time with $15 \sim 80$ frame rates in the system.

Experiment 6.1 uses the proposed system to calculate a lifting path for a user defined lifting task in the point cloud environment. The lifting task requires the tower crane to lift the target from one side of box 1 to another location and place it onto the



(a) (b) (c)

(d) (e) (f)

(g) (h)

Figure 6.13: Lifting path simulated using point cloud data in the proposed system.

(a) (b) (c)

(d) (e) (f)

(g) (h)

Figure 6.14: Lifting path conducted by the scaled model crane.

top of box 2. During the process, the lifting target should also be kept at a clearance from all the obstacles. It takes the system 584 ms to rasterize the point cloud and 2031 ms to calculate a collision-free lifting path. The lifting path (Table 6.1 and Figure 6.12) contains 8 steps, conducting totally 65 degrees of swinging, 14.53 centimeters of trolley movements and 21.88 centimeters of hoisting. Figure 6.13 demonstrates the key-frame configurations of the ideal lifting path in the simulation environment. The point cloud is simplified in the figure to show the path more clearly.

After the planning stage is finished, the lifting plan is exported and sent to the physical crane to be automatically conducted. The real lifting path conducted by the model crane is shown in Figure 6.14. The model crane is able to conduct the path generated by the system to move the lifting target following the trajectory as displayed in the simulation (Figure 6.13). No collision happens during the entire lifting process.

Due to the difficulty to have an actual crane for this research including this experiment, Experiment 6.1 is conducted on a model crane and a room environment. Although they are much smaller in size than those in real lifting scenarios, the complexity of the problem is comparable to the real cases. This complexity can be observed from the number of points in the point cloud environments. The number of points in Experiment 6.1 (21,000,716 points) is even larger than that in Experiment 6.2 (13,341,982 points) which is from a real industrial plant. Therefore, the results of Experiment 6.1 show that it is promising to use the proposed system for real lifting applications. Experiment 6.1 and its results have proven that the proposed system is able to fulfill the entire workflow of automatic lifting and could perform it efficiently and accurately.

6.3.2 Case study of offline lift planning in industrial projects

To demonstrate the application of the system for off-line lift planning, a case study is conducted in an industrial plant in Singapore. A Terex AC435 terrain crane is parked in the center of the plant to lift a bundle of pipes. In this case study, the proposed system is used as a planner to produce an optimal single-crane lifting path for this specific crane and industrial site. The path is then used to guide the operations of the physical crane on the same site.

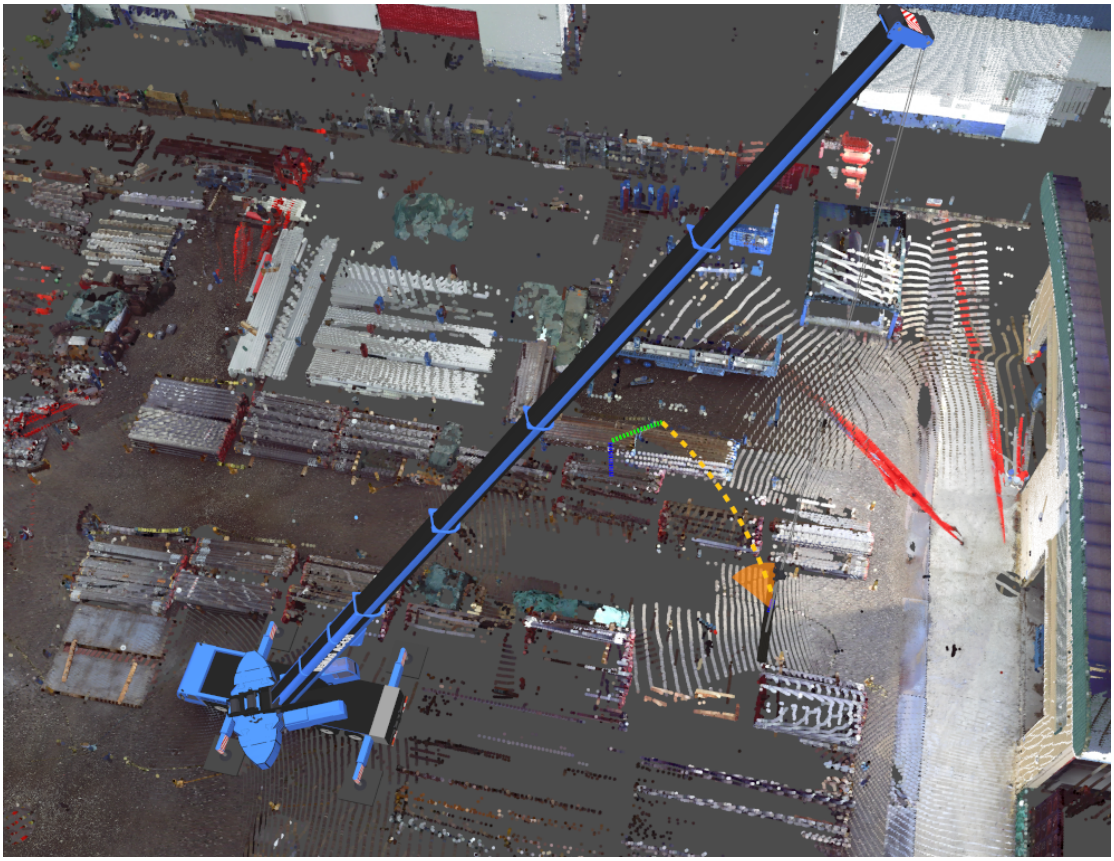


Figure 6.15: Lifting path generated by the system using an accurate digital version of the crane and the scanned point cloud of the site.

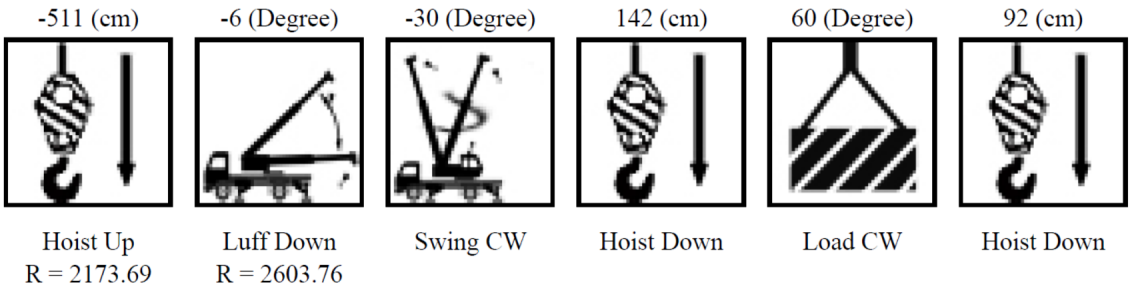


Figure 6.16: The lifting path documentation for guidance of lifting operations.

The planning starts from taking laser scans of the site. Nine scans are taken at the site within two days. Data from these scans are then registered as a single point cloud and sub-sampled by a rate of 1/9. The final point cloud data set contains 13,341,982 points. The digital version of the Terex AC435 terrain crane is designed according

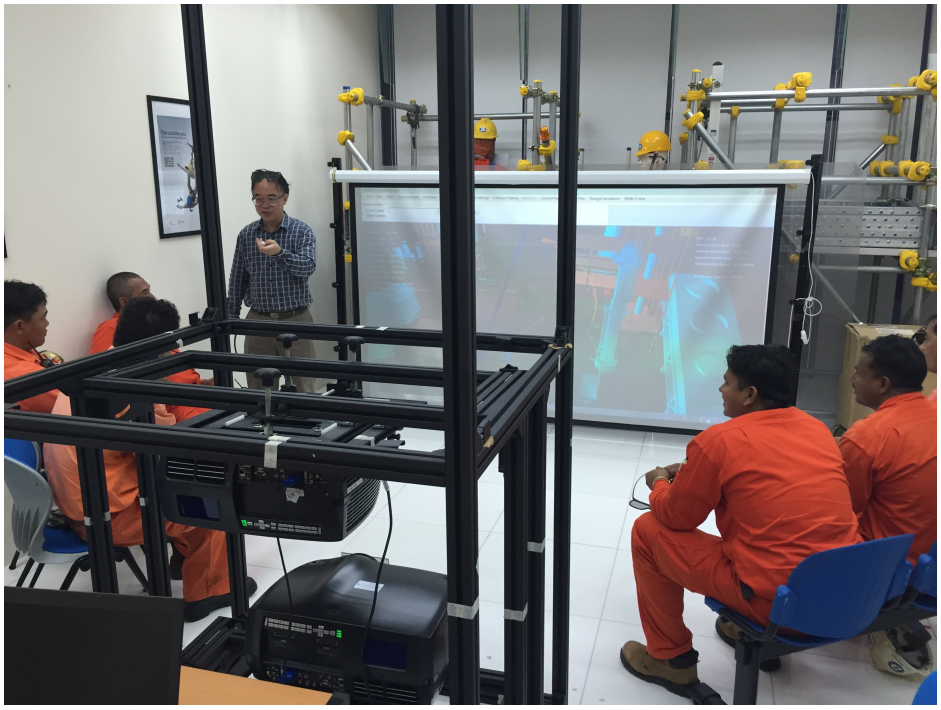
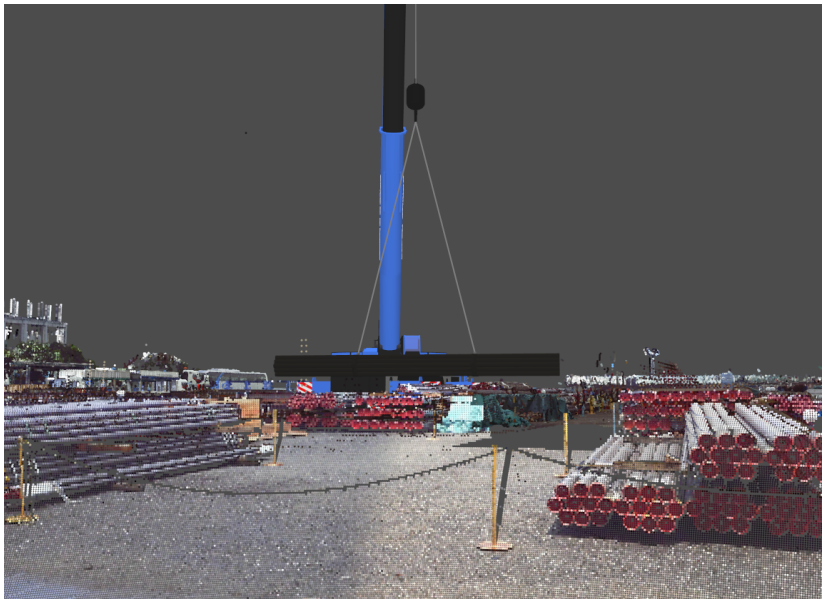


Figure 6.17: The proposed system used for communication in the lifting team.

to its actual sizes and structures, and loaded in to the crane database of the system. The digital crane is located in the virtual scene as where the physical crane is on the site. This is fulfilled by matching the digital crane with the scanned point cloud of the physical crane which has already been settled on the site before the laser scans. For other cases where the crane is not present during scanning, markers can also be used to help record its setup location. Automatic lifting path planning is then performed using user inputs such as crane setup details and pick & place locations. The lifting path output from the automatic planning engine is reviewed and validated in the simulation environment (Figure 6.15). Data of the optimal path are then exported as a lifting plan document clearly stating the operation sequence and movement amounts (Figure 6.16) for the guidance of lifting operations. Before starting the real lifting process, the lifting operations are briefed to the lifting team with simulations and animations (Figure 6.17). Finally, the real lifting is conducted following the guidance of the lift plan documentation (Figure 6.18). This case study shows that, the proposed system is feasible and provides a complete solution for off-line lift planning.

(a)



(b)

Figure 6.18: Lifting conducted using the suggestions from the system: (a) real lifting operations; (b) simulated lifting using point cloud data in the proposed system.

6.3.3 Training and interactive lift planning using 3D simulation

Figure 6.19 shows the scenario when the system is used as a simulator where the user manipulates the virtual crane with a joystick. When the crane is moving, the

Figure 6.19: The proposed system used for operator training and interactive lift planning purposes with joystick interactions and stereoscopic display.

manipulation parameters are also updated on the GUI. The user wears a pair of 3D glasses to perceive the stereoscopically rendered lifting scene. When trying to fulfill a task in the simulation environment, invalid behaviors of the user will trigger safety warnings. In this way, the user is trained to perform safe lifting operations. On the other hand, the operation steps conducted by the user are recorded. These user-defined lifting paths can be evaluated by the fitness functions to check their safety factors and optimally. Users are also allowed to modify the paths according to the evaluation results and output them as plans. In this case, the system also serves interactive lift planning purpose.

6.4 Summary

A lift planner cum crane simulator system making use of PDMS, BIM, Smartplant, and laser scanning technologies has been designed to help industries improve safety and efficiency in lifting. The system has unified various types of environment data and

embedded the technologies introduced in the previous chapters to serve interactive and automatic lift planning purpose. Using accurate modeling of industrial plants and sites, the system is able to produce path suggestions for the guidance of real-life lifting tasks. It also enables better training for crane operators to practice safe lifting operations in realistic virtual environments.

Chapter 7

Conclusions and Future Work

7.1 Summary of the Research

The research has solved two path planning problems regarding the lifting operations of single and cooperative (dual) cranes. An efficient and scalable image-space collision detection algorithm has been developed to deal with complex environments. Based on this collision detection algorithm, two efficient MSPGA-based path planners for single-crane and dual-crane lifting have been prototyped. Problem-specific fitness functions and genetic operators have been designed using an LGP strategy. These planners use ASVs and TSSs to perform CCD and employ a hybrid C-space collision detection strategy to reduce the planning time.

The MSPGA-based path planner for single-crane lifting can produce short, safe and easy-to-conduct lifting paths in near real-time performance for any complex environments. Compared to previous methods, the planner for dual-crane lifting can better handle the cooperation between the two cranes and balance of the cable-suspended lifting target. The planner developed has also achieved good search abilities with high success rates in complex industrial plants and sites. Finally, the proposed algorithms have been developed into a lift planner cum crane simulator which enables both interactive and automatic lift planning. By combining with laser scanning, PDMS, Smartplant and BIM technologies, the system has been designed to serve as a practical tool for automatic lift planning and vocational training.

In summary, novelties of the proposed solution in this research are listed below:

(i) A unified MDM representation has been proposed and implemented for complex industrial plants and sites. This representation, combined with OBBs, ASVs and TSSs, enables efficient collision detection for the crane components and the lifting target.

(ii) Comprehensive mathematical formulations considering practical issues in single-crane and dual-crane lifting path planning have been developed.

(iii) LGP enhanced MSPGA-based path planners with customized adaptive plans and multi-level GPU parallelization have been designed and developed.

(iv) A hybrid C-space collision strategy has been proposed and incorporated into the planners to achieve optimal computational performance.

(v) A TSS-based swept volume for continuous collision detection has been proposed and integrated.

(vi) A novel prototype system for lift planning and crane simulation which supports various formats of digital plants and sites has been designed, implemented and initially validated.

7.2 Limitations and Future Work

The potentials of the proposed algorithms are not fully exploited yet due to the time constraints. The paths output by the planners are more suitable for off-line lift planning. To achieve autonomous control of cranes, dynamics of the cranes shall be included to generate trajectories with time information. The collision detection algorithm proposed in Chapter 3 only considers static environments with one or two cranes interacting with them. To make the algorithm more general, dynamic environments need to be considered. For the proposed path planning algorithms, handling dynamic environments requires a re-planning mechanism to fast produce revised lifting paths according to the partially changed environments. Moreover, the proposed path planning algorithms only consider mobile cranes without jibs. The ASVs and hybrid C-spaces need to be generalized before being applied to crawler cranes with jibs or other crane-like

robots. In some occasions, one may use more than two cranes to lift a heavy load cooperatively. Thus, the existing dual-crane planner needs to be generalized for multi-crane occasions. To address these limitations, the future work of the research includes the following aspects:

(i) **Trajectory generation from output lifting paths.** Trajectories with time information are necessary for automatic control of cranes. The trajectories are required to contain smooth movements. Velocities and accelerations might also be constrained due to mechanical limitations. For the lifting problem, trajectories can be built from the lifting paths by assigning time information to the paths according to the estimated speed of operations. Then, the key frame configurations can be interpolated or approximated by a smooth trajectory as a general or rational polynomial curve. The trajectories may also need to be further smoothed according to the dynamic constraints.

(ii) **Adding re-planning mechanisms to deal with dynamic environments.** The optimal paths need to be modified when the environment has changed. This requirement comes up in many occasions. For example, when finishing a lifting task in a construction site, the detached lifting target becomes a new obstacle in the scene. This also happens when objects are removed or added in the plant after laser scans have been conducted. In this case, the previous plan can be reused to fast generate a new plan for the updated point cloud. The re-planning functionality can be achieved by using the GA population obtained at the last termination as the initial population for the new GA search. However, the existing population might have been conquered by several elite paths and are not able to provide enough diversity for future searches. In this case, new random chromosomes should be generated and used to replace a portion of the duplicated elite paths in the population. In order to make the random seeds survive, the selection pressure should be decreased in the starting phases.

(iii) **Supporting more types of cranes.** To handle crawler cranes with jibs and other types of crane-like robots, more general CCD methods need to be applied.

However, it is difficult to achieve accurate CCD within reasonable execution time and with balanced load for GPU parallelization. A possible way is to represent the swept volumes of the booms and jibs as SSVs. For single-crane lifting, the centroid primitives can be analytically represented based on forward kinematics since the operations are decoupled. For dual-crane lifting, as the cranes perform small-scale movements between consecutive steps, the RSSs can be chosen as the SVs.

(iv) **Cooperative lifting with more than two cranes.** The complexity of the lifting path planning problem increases exponentially with the number of cranes involved. The balancing equations of the suspension system need to be reformulated to support a larger number of cranes. When the number of cooperative cranes equals to three, twelve equations can be formulated taking into consideration of the force and torque balancing, the shape of the target and crane-target linkages. The same number of variables can also be defined by including six sling angles, three sling lengths and the mass center of the lifting target. However, the equation system will be overdetermined if the number of cranes exceeds three. In these cases, three featuring cranes can be selected to determine the location and orientation of the target. The motion of the lifting target can be handled similarly in path planning by using SSVs with uncertainty thresholds. In these cases, the centroid primitives in SSVs will be the polygons.

(v) **Improving the support of big geometric data.** The amount of data acquired from the PDMS, Smartplant, BIM and laser scanning technologies can be huge, especially for accurately modeling large and complex industrial plants or sites. Handling the big data sets brings challenges to the proposed system. A possible way to handle these data would be utilizing the idea of LOD for rendering and MDM generations. This idea can be implemented by constructing quadtrees for the geometric data sets in the x-y plane and organizing the depth information according to quadtrees. In this way, the load of handling large data sets can be shifted to the pre-processing stage where the quadtrees are constructed and MDMs for leaf nodes are generated.

(vi) **Improving GPU parallelization.** The performance speedups brought by GPUs can be further exploited. The usages of different memory types need to be balanced in order to minimize the data transfer time. The communication frequency between the CPU and GPU are to be minimized to reduce the latency in simulation and the execution time of path planning. Global memories need to be aligned properly to maximize the concurrency in memory accessing.

(vii) **Further validations of the lift planner cum crane simulator system.** The experiment presented in this thesis is a preliminary validation on the system. More case studies will be conducted for more complex environments, for cooperative dual-crane lifting tasks, and for dynamic scenes.

Publications

Patent:

Y. Cai, P. Cai, C. Indhumathi, J. Zheng, N. M. Thalmann, P. Wong, T. S. Lim and Y. Gong, "Method and system for intelligent crane lifting", PCT filing (US-A/German/China/Singapore) PCT/SG2014/000472, 8 October 2014.

Journal papers[1]:

P. Cai, Y. Cai, I. Chandrasekaran, and J. Zheng, "Parallel GA based automatic crane lifting path planning in complex environments", Automation in Construction (5-year impact factor: 2.414), Volume 62, February 2016, Pages 133-147, ISSN 0926-5805, http://dx.doi.org/10.1016/j.autcon.2015.09.007.

P. Cai, Y. Cai, I. Chandrasekaran, and J. Zheng, "Automatic path planning for dual-crane lifting in complex environments using LGP-enhanced parallel genetic algorithm", under review by Automation in Construction.

Conference papers:

P. Cai, C. Indhumathi, Y. Cai, J. Zheng, "A Framework of the crane simulator using GPU-based collision detection", Workshop on Serious Game & Simulation, CASA, May 2012, Singapore.

P. Cai, Y. Cai, I. Chandrasekaran, and J. Zheng, "A GPU-enabled parallel genetic algorithm for path planning", 2013 Symposium on GPU Computing and Applications, Oct 2013, Singapore. (*Best Paper Award*)

[1]The 5-year impact factors are obtained from the 2014 Journal Citation Reports.

P. Cai, C. Indhumathi, and Y. Cai, "High performance simulation for vocational training of heavy mobile cranes", the Second Asia-Europe Symposium on Simulation and Serious Games, October 2014, Zwolle, The Netherlands.

Book chapters:

P. Cai, C. Indhumathi, Y. Cai, J. Zheng, Y. Gong, T. Lim, and P. Wong, "Collision detection using axis aligned bounding boxes", in Simulations, Serious Games and Their Applications (Y. Cai and S. L. Goei, eds.), Gaming Media and Social Effects, pp. 1-14, Springer Singapore, 2014.

P. Cai, Y. Cai, I. Chandrasekaran, and J. Zheng, "A gpu-enabled parallel genetic algorithm for path planning of robotic operators", in Y. Cai and Simon See (eds.), GPU Computing and Applications, pp. 1-13, Springer, 2015.

References

[1] M. A. D. Ali, N. R. Babu, and K. Varghese, "Collision free path planning of cooperative crane manipulators using genetic algorithm," *Journal of Computing in Civil Engineering*, vol. 19, no. 2, pp. 182–193, 2005.

[2] Y. C. Chang, W. H. Hung, and S. C. Kang, "A fast path planning method for single and dual crane erections," *Automation in Construction*, vol. 22, pp. 468–480, 2012.

[3] S. M. LaValle, *Planning Algorithms*. New York, NY, USA: Cambridge University Press, 2006.

[4] OSHA, "Industry injury and illness data." http://www.bls.gov/iif/oshsum.htm, 2013. Online: accessed 19-Dec-2013.

[5] A. Ltd, "A guide to rental rate in 2011." http://www.vertikal.net/uploads/tx_filelinks/ca_2011_9_p16-27.pdf, 2011. Online: accessed 19-July-2014.

[6] Liebherr, "Product advantages - mobile crane LTM 1200-5.1." http://www.liebherr.com/AT/en-GB/products_at.wfw/id42940/measure-metric/tab3742_1477, 2007. Online: accessed on 21-JUL-2015.

[7] Q. Avril, V. Gouranton, and B. Arnaldi, "New trends in collision detection performance," *VRIC'09 Proceedings*, vol. 11, 2009.

[8] K. Ward, F. Bertails, T. Y. Kim, S. R. Marschner, M. P. Cani, and M. C. Lin, "A survey on hair modelling: Styling, simulation, and rendering," *IEEE Transactions on Visualization and Computer Graphics*, vol. 13, no. 2, pp. 213–234, 2007.

[9] B. Thomaszewski, S. Pabst, and W. Blochinger, "Parallel techniques for physically based simulation on multi-core processor architectures," *Computers & Graphics*, vol. 32, no. 1, pp. 25–40, 2008.

[10] Y. Tsuji, T. Kawaguchi, and T. Tanaka, "Discrete particle simulation of two-dimensional fluidized bed," *Powder technology*, vol. 77, no. 1, pp. 79–87, 1993.

[11] NVIDIA Corporation, "CUDA code samples." https://developer.nvidia.com/cuda-code-samples, 2007. Online: accessed on 21-JUL-2015.

[12] NVIDIA Corporation, "Fermi hair demo." http://www.geforce.com/games-applications/pc-applications/fermi-hair-demo, 2010. Online: accessed on 21-JUL-2015.

[13] T. N. A. Administration and Space, "Nasa homepage," 2015.

[14] B. G. Baumgart, "A polyhedron representation for computer vision," in *Proceedings of the May 19-22, 1975, National Computer Conference and Exposition*, AFIPS '75, (New York, NY, USA), pp. 589–596, ACM, 1975.

[15] M. McGuire, "The half-edge data structure." http://www.flipcode.com/archives/The_Half-Edge_Data_Structure.shtml, 2000. Online: accessed 17-April-2015.

[16] N. Govindaraju, I. Kabul, M. Lin, and D. Manocha, "Fast continuous collision detection among deformable models using graphics processors," *Computers & Graphics*, vol. 31, no. 1, pp. 5–14, 2007.

[17] C. Ericson, *Real-Time Collision Detection (The Morgan Kaufmann series in Interactive 3-D Technology)*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2004.

[18] X. Provot, "Collision and self-collision handling in cloth model dedicated to design garments," in *Computer Animation and Simulation '97: Proceedings of the Eurographics Workshop* (D. Thalmann and M. Panne, eds.), (Vienna), pp. 177–189, Springer Vienna, September 1997.

[19] S. K. Wong, C. M. Liu, G. Baciu, and C. C. Yeh, "Robust continuous collision detection for deformable objects," in *Proceedings of the 17th ACM Symposium on Virtual Reality Software and Technology*, VRST '10, (New York, NY, USA), pp. 55–62, ACM, 2010.

[20] S. Gottschalk, M. C. Lin, and D. Manocha, "OBBTree: A hierarchical structure for rapid interference detection," in *Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '96, (New York, NY, USA), pp. 171–180, ACM, 1996.

[21] G. van den Bergen, "Efficient collision detection of complex deformable models using AABB trees," *J. Graph. Tools*, vol. 2, pp. 1–13, Jan. 1998.

[22] D. Baraff, *Dynamic Simulation of Non-penetrating Rigid Bodies*. PhD thesis, Cornell University, Ithaca, NY, USA, 1992. PhD thesis.

[23] C. Shaffer and G. Herb, "A real-time robot arm collision avoidance system," *IEEE Transactions on Robotics and Automation*, vol. 8, pp. 149–160, Apr 1992.

[24] B. Faverjon, "Hierarchical object models for efficient anti-collision algorithms," in *Proceedings of IEEE International Conference on Robotics and Automation*, pp. 333–340 vol.1, May 1989.

[25] J. D. Cohen, M. C. Lin, D. Manocha, and M. Ponamgi, "I-collide: An interactive and exact collision detection system for large-scale environments," in *Proceedings of the 1995 Symposium on Interactive 3D Graphics*, I3D '95, (New York, NY, USA), pp. 189–196, ACM, 1995.

[26] T. C. Hudson, M. C. Lin, J. Cohen, S. Gottschalk, and D. Manocha, "V-COLLIDE: accelerated collision detection for vrml," in *Proceedings of the second symposium on Virtual reality modeling language*, pp. 117–ff, ACM, 1997.

[27] K. Zhou, M. Gong, X. Huang, and B. Guo, "Data-parallel octrees for surface reconstruction," *IEEE Transactions on Visualization and Computer Graphics*, vol. 17, no. 5, pp. 669–681, 2011.

[28] D. Jung and K. K. Gupta, "Octree-based hierarchical distance maps for collision detection," in *Proceedings of 1996 IEEE International Conference on Robotics and Automation*, vol. 1, pp. 454–459, IEEE, Apr 1996.

[29] B. Von Herzen, A. H. Barr, and H. R. Zatz, "Geometric collisions for time-dependent parametric surfaces," *SIGGRAPH Comput. Graph.*, vol. 24, pp. 39–48, Sept. 1990.

[30] S. Ar, B. Chazelle, and A. Tal, "Self-customized BSP trees for collision detection," *Computational Geometry*, vol. 15, no. 13, pp. 91 – 102, 2000.

[31] P. Hubbard, "Interactive collision detection," in *In Proceedings of IEEE Symposium on Research Frontiers in Virtual Reality*, pp. 24–31, 1993.

[32] X. Zhang, Y. J. Kim, and D. Manocha, "Continuous penetration depth," *Comput. Aided Des.*, vol. 46, pp. 3–13, Jan. 2014.

[33] G. Van den Bergen, "A fast and robust GJK implementation for collision detection of convex objects," *J. Graph. Tools*, vol. 4, pp. 7–25, Mar. 1999.

[34] X. Zhang, S. Redon, M. Lee, and Y. J. Kim, "Continuous collision detection for articulated models using Taylor models and temporal culling," *ACM Trans. Graph.*, vol. 26, July 2007.

[35] J. Klosowski, M. Held, J. Mitchell, H. Sowizral, and K. Zikan, "Efficient collision detection using bounding volume hierarchies of k-DOPs," *IEEE Transactions on Visualization and Computer Graphics*, vol. 4, pp. 21–36, Jan 1998.

[36] S. K. Wong, W. C. Lin, C. H. Hung, Y. J. Huang, and S. Y. Lii, "Radial view based culling for continuous self-collision detection of skeletal models," *ACM Trans. Graph.*, vol. 32, pp. 114:1–114:10, July 2013.

[37] M. Tang, S. Curtis, S.-E. Yoon, and D. Manocha, "ICCD: Interactive continuous collision detection between deformable models using connectivity-based culling," *IEEE Transactions on Visualization and Computer Graphics*, vol. 15, pp. 544–557, July 2009.

[38] E. Larsen, S. Gottschalk, M. C. Lin, and D. Manocha, "Fast proximity queries with swept sphere volumes," tech. rep., Technical Report TR99-018, Department of Computer Science, University of North Carolina, 1999.

[39] S. Redon, Y. J. Kim, M. C. Lin, and D. Manocha, "Fast continuous collision detection for articulated models," in *Proceedings of the Ninth ACM Symposium on Solid Modeling and Applications*, SM '04, (Aire-la-Ville, Switzerland), pp. 145–156, Eurographics Association, 2004.

[40] J. Corrales, F. Candelas, and F. Torres, "Safe humanrobot interaction based on dynamic sphere-swept line bounding volumes," *Robotics and Computer-Integrated Manufacturing*, vol. 27, no. 1, pp. 177 – 185, 2011.

[41] C. Lauterbach, Q. Mo, and D. Manocha, "gProximity: Hierarchical GPU-based operations for collision and distance queries," *Computer Graphics Forum*, vol. 29, no. 2, pp. 419–428, 2010.

[42] D. Kim, J.-P. Heo, J. Huh, J. Kim, and S.-e. Yoon, "HPCCD: Hybrid parallel continuous collision detection using CPUs and GPUs," *Computer Graphics Forum*, vol. 28, no. 7, pp. 1791–1800, 2009.

[43] M. Tang, D. Manocha, and R. Tong, "MCCD: Multi-core collision detection between deformable models using front-based decomposition," *Graphical Models*, vol. 72, no. 2, pp. 7–23, 2010.

[44] M. Teschner, S. Kimmerle, B. Heidelberger, G. Zachmann, L. Raghupathi, A. Fuhrmann, M.-P. Cani, F. Faure, N. Magnenat-Thalmann, W. Strasser, and P. Volino, "Collision detection for deformable objects," *Computer Graphics Forum*, vol. 24, no. 1, pp. 61–81, 2005.

[45] M. Shinya and M.-C. Forgue, "Interference detection through rasterization," *The Journal of Visualization and Computer Animation*, vol. 2, no. 4, pp. 132–134, 1991.

[46] G. Baciu, W. Wong, and H. Sun, "Recode: an image-based collision detection algorithm," *The Journal of Visualization and Computer Animation*, vol. 10, pp. 181–192, 1999.

[47] B. Heidelberger, M. Teschner, and M. H. Gross, "Real-time volumetric intersections of deforming objects," in *Proceedings of Vision, Modeling, and Visualization*, vol. 3, pp. 461–468, AKA, 2003.

[48] B. Heidelberger, M. Teschner, and M. Gross, "Detection of collisions and self-collisions using image-space techniques," *Journal of WSCG*, vol. 12, no. 3, pp. 145–152, 2004.

[49] F. Faure, S. Barbier, J. Allard, and F. Falipou, "Image-based collision detection and response between arbitrary volume objects," in *Proceedings of the 2008 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, SCA '08, (Aire-la-Ville, Switzerland), pp. 155–162, Eurographics Association, 2008.

[50] N. K. Govindaraju, S. Redon, M. C. Lin, and D. Manocha, "CULLIDE: Interactive collision detection between complex models in large environments using graphics hardware," in *Proceedings of the ACM SIGGRAPH/EUROGRAPHICS Conference on Graphics Hardware*, HWWS '03, (Aire-la-Ville, Switzerland), pp. 25–32, Eurographics Association, 2003.

[51] N. Govindaraju, M. Lin, and D. Manocha, "Quick-CULLIDE: fast inter- and intra-object collision culling using graphics hardware," in *Virtual Reality. Proceedings of VR 2005*, pp. 59–66, IEEE, March 2005.

[52] NVIDIA Corporation, "NVIDIA CUDA C programming guide," 2010. Version 3.2.

[53] J. Sanders and E. Kandrot, *CUDA by example: an introduction to general-purpose GPU programming*. Addison-Wesley Professional, 2010.

[54] A. Greß and G. Zachmann, "Object-space interference detection on programmable graphics hardware," in *SIAM Conf. on Geometric Design and Computing*, pp. 311–328, 2003.

[55] J. Pan and D. Manocha, "GPU-based parallel collision detection for fast motion planning," *Int. J. Rob. Res.*, vol. 31, pp. 187–200, Feb. 2012.

[56] S. J. Guy, J. Chhugani, C. Kim, N. Satish, M. Lin, D. Manocha, and P. Dubey, "Clearpath: highly parallel collision avoidance for multi-agent simulation," in *Proceedings of the 2009 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, SCA '09, (New York, NY, USA), pp. 177–187, ACM, 2009.

[57] M. Tang, D. Manocha, J. Lin, and R. Tong, "Collision-streams: fast GPU-based collision detection for deformable models," in *Symposium on interactive 3D graphics and games*, I3D '11, (New York, NY, USA), pp. 63–70, ACM, ACM, 2011.

[58] J. H. Holland, *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control and Artificial Intelligence*. Cambridge, MA, USA: MIT Press, 1992.

[59] J. Holland, "Genetic algorithms," *Scientific American*, vol. 267, no. 1, pp. 66–72, 1992.

[60] D. E. Goldberg, *Genetic algorithms in search, optimization, and machine learning*, vol. 412. Addison-wesley Reading Menlo Park, 1989.

[61] W. M. Spears and K. A. De Jong, "An analysis of multi-point crossover," tech. rep., DTIC Document, 1990.

[62] K. A. De Jong, *An Analysis of the Behavior of a Class of Genetic Adaptive Systems.* PhD thesis, University of Michigan, Ann Arbor, MI, USA, 1975. AAI7609381.

[63] Z. Michalewicz and G. Nazhiyath, "Genocop III: A co-evolutionary algorithm for numerical optimization problems with nonlinear constraints," in *IEEE International Conference on Evolutionary Computation*, vol. 2, pp. 647–651, IEEE, Nov 1995.

[64] J. Biegel and J. Davern, "Genetic algorithms and job shop scheduling," *Computers & Industrial Engineering*, vol. 19, no. 1, pp. 81–91, 1990.

[65] P. Poon and J. Carter, "Genetic algorithm crossover operators for ordering applications," *Computers & Operations Research*, vol. 22, no. 1, pp. 135–147, 1995.

[66] A. Hamidinia, S. Khakabimamaghani, M. Mazdeh, and M. Jafari, "A genetic algorithm for minimizing total tardiness/earliness of weighted jobs in a batched delivery system," *Computers & Industrial Engineering*, vol. 62, no. 1, pp. 29–38, 2012.

[67] D. E. Goldberg, "Simple genetic algorithms and the minimal, deceptive problem," in *Genetic Algorithms and Simulated Annealing* (L. Davis, ed.), Research Notes in Artificial Intelligence, pp. 74–88, Pitman, 1987.

[68] D. E. Goldberg and K. Deb, "A comparative analysis of selection schemes used in genetic algorithms," *Foundations of genetic algorithms*, vol. 1, pp. 69–93, 1991.

[69] B. Miller and D. Goldberg, "Genetic algorithms, selection schemes, and the varying effects of noise," *Evolutionary Computation*, vol. 4, no. 2, pp. 113–131, 1996.

[70] J. Smith and T. C. Fogarty, "Self adaptation of mutation rates in a steady state genetic algorithm," in *Proceedings of IEEE International Conference on Evolutionary Computation*, pp. 318–323, IEEE, 1996.

[71] D. Bhandari, C. Murthy, and K. Sankar, "Genetic algorithm with elitist model and its convergence," *International Journal of Pattern Recognition and Artificial Intelligence*, vol. 10, no. 6, pp. 731–747, 1996.

[72] B. Chen, Y. Cheng, and C. Lee, "A genetic approach to mixed h^2/h^∞ optimal pid control," *Control Systems, IEEE*, vol. 15, no. 5, pp. 51–60, 1995.

[73] J. Renders and S. Flasse, "Hybrid methods using genetic algorithms for global optimization," *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, vol. 26, no. 2, pp. 243–258, 1996.

[74] M. Safe, J. Carballido, I. Ponzoni, and N. Brignole, "On stopping criteria for genetic algorithms," in *Advances in Artificial Intelligence SBIA 2004* (A. Bazzan and S. Labidi, eds.), vol. 3171 of *Lecture Notes in Computer Science*, pp. 405–413, Springer Berlin Heidelberg, 2004.

[75] S. W. Mahfoud, "Population size and genetic drift in fitness sharing," in *Proceedings of the Third Workshop on Foundations of Genetic Algorithms. Estes Park, Colorado, USA, July 31 - August 2 1994*, pp. 185–223, 1994.

[76] C. Fonseca and P. Fleming, "Multiobjective genetic algorithms made easy: selection sharing and mating restriction," in *Genetic Algorithms in Engineering Systems: Innovations and Applications. First International Conference on GALESIA. (Conf. Publ. No. 414)*, pp. 45–52, Sep 1995.

[77] M. A. Ismail, "Parallel genetic algorithms (pgas): master slave paradigm approach using mpi," in *E-Tech 2004*, pp. 83–87, IEEE, July 2004.

[78] T. h. Zhao, Z. b. Man, and X. y. Qi, "A MSM-PGA based on multi-agent and its application in reactive power optimization of power systems," in *Third International Conference on Electric Utility Deregulation and Restructuring and Power Technologies*, pp. 2037–2041, 2008.

[79] B. Skinner, H. Nguyen, and D. Liu, "Performance study of a multi-deme parallel genetic algorithm with adaptive mutation," in *Proceedings of the 2nd International Conference on Autonomous Robots and Agents*, 2004.

[80] J. Wakeley, "The coalescent in an island model of population subdivision with variation among demes," *Theoretical population biology*, vol. 59, no. 2, pp. 133–144, 2001.

[81] B. Manderick and P. Spiessens, "Fine-grained parallel genetic algorithms," in *Proceedings of the 3rd International Conference on Genetic Algorithms*, (San Francisco, CA, USA), pp. 428–433, Morgan Kaufmann Publishers Inc., 1989.

[82] H. Mühlenbein, M. Schomisch, and J. Born, "The parallel genetic algorithm as function optimizer," *Parallel computing*, vol. 17, no. 6, pp. 619–632, 1991.

[83] E. W. Dijkstra, "A note on two problems in connexion with graphs," *Numer. Math.*, vol. 1, pp. 269–271, Dec. 1959.

[84] P. Hart, N. Nilsson, and B. Raphael, "A formal basis for the heuristic determination of minimum cost paths," *IEEE Transactions on Systems Science and Cybernetics*, vol. 4, pp. 100–107, July 1968.

[85] A. Soltani, H. Tawfik, J. Goulermas, and T. Fernando, "Path planning in construction sites: performance evaluation of the Dijkstra, A*, and GA search algorithms," *Advanced Engineering Informatics*, vol. 16, no. 4, pp. 291 – 303, 2002.

[86] R. Kala, A. Shukla, R. Tiwari, S. Rungta, and R. Janghel, "Mobile robot navigation control in moving obstacle environment using genetic algorithm, artificial neural networks and A* algorithm," in *WRI World Congress on Computer Science and Information Engineering*, vol. 4, pp. 705–713, March 2009.

[87] R. Kala, A. Shukla, and R. Tiwari, "Fusion of probabilistic A* algorithm and fuzzy inference system for robotic path planning," *Artif. Intell. Rev.*, vol. 33, pp. 307–327, Apr. 2010.

[88] A. Stentz, "Optimal and efficient path planning for partially-known environments," in *Proceedings of IEEE International Conference on Robotics and Automation*, pp. 3310–3317 vol.4, May 1994.

[89] S. Koenig, M. Likhachev, and D. Furcy, "Lifelong planning A*," *Artificial Intelligence*, vol. 155, no. 12, pp. 93 – 146, 2004.

[90] S. Koenig and M. Likhachev, "Fast replanning for navigation in unknown terrain," *IEEE Transactions on Robotics*, vol. 21, pp. 354–363, June 2005.

[91] J. Barraquand and J.-C. Latombe, "A Monte-Carlo algorithm for path planning with many degrees of freedom," in *Proceedings of IEEE International Conference on Robotics and Automation*, pp. 1712–1717 vol.3, May 1990.

[92] L. Kavraki, P. Svestka, J.-C. Latombe, and M. Overmars, "Probabilistic roadmaps for path planning in high-dimensional configuration spaces," *IEEE Transactions on Robotics and Automation*, vol. 12, pp. 566–580, Aug 1996.

[93] S. M. LaValle, J. J. Kuffner, and Jr., "Rapidly-exploring random trees: Progress and prospects," 2000. Technical report.

[94] J. Barraquand and J.-C. Latombe, "Robot motion planning: A distributed representation approach," *Int. J. Rob. Res.*, vol. 10, pp. 628–649, Dec. 1991.

[95] B. Krogh and C. Thorpe, "Integrated path planning and dynamic steering control for autonomous vehicles," in *Proceedings of IEEE International Conference on Robotics and Automation*, vol. 3, pp. 1664–1669, Apr 1986.

[96] Y. Koren and J. Borenstein, "Potential field methods and their inherent limitations for mobile robot navigation," in *Proceedings of IEEE International Conference on Robotics and Automation*, pp. 1398–1404 vol.2, Apr 1991.

[97] S. Caselli, M. Reggiani, and R. Rocchi, "Heuristic methods for randomized path planning in potential fields," in *Proceedings of 2001 IEEE International Symposium on Computational Intelligence in Robotics and Automation*, pp. 426–431, 2001.

[98] L. Kavraki, M. Kolountzakis, and J.-C. Latombe, "Analysis of probabilistic roadmaps for path planning," *IEEE Transactions on Robotics and Automation*, vol. 14, pp. 166–171, Feb 1998.

[99] S. Karaman and E. Frazzoli, "Sampling-based algorithms for optimal motion planning," *Int. J. Rob. Res.*, vol. 30, pp. 846–894, June 2011.

[100] D. Hsu, J. C. Latombe, and H. Kurniawati, "On the probabilistic foundations of probabilistic roadmap planning," *International Journal of Robotics Research*, vol. 25, no. 7, pp. 627–643, 2006.

[101] R. Bohlin and L. Kavraki, "Path planning using lazy PRM," in *IEEE International Conference on Robotics and Automation. Proceedings of ICRA '00*, vol. 1, pp. 521–528 vol.1, April 2000.

[102] C. Nielsen and L. Kavraki, "A two level fuzzy PRM for manipulation planning," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2000)*, vol. 3, pp. 1716–1721 vol.3, Oct 2000.

[103] G. Sánchez and J.-C. Latombe, "On delaying collision checking in prm planning - application to multi-robot coordination," *International Journal of Robotics Research*, vol. 21, no. 1, pp. 5–26, 2002.

[104] J. Denny and N. M. Amato, "Toggle PRM: Simultaneous mapping of c-free and c-obstacle - a study in 2d -," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 2632–2639, 2011.

[105] J. D. Marble and K. E. Bekris, "Asymptotically near-optimal planning with probabilistic roadmap spanners," *IEEE Transactions on Robotics*, vol. 29, no. 2, pp. 432–444, 2013.

[106] J. Kuffner and S. LaValle, "RRT-connect: An efficient approach to single-query path planning," in *IEEE International Conference on Robotics and Automation. Proceedings of ICRA '00*, vol. 2, pp. 995–1001 vol.2, 2000.

[107] S. Karaman and E. Frazzoli, "Optimal kinodynamic motion planning using incremental sampling-based methods," in *49th IEEE Conference on Decision and Control (CDC)*, pp. 7681–7687, Dec 2010.

[108] S. Karaman, M. Walter, A. Perez, E. Frazzoli, and S. Teller, "Anytime motion planning using the RRT*," in *IEEE International Conference on Robotics and Automation (ICRA)*, pp. 1478–1483, May 2011.

[109] I. Ko, B. Kim, and F. C. Park, "Randomized path planning on vector fields," *Int. J. Rob. Res.*, vol. 33, pp. 1664–1682, Nov. 2014.

[110] M. Otte and E. Frazzoli, "RRTX: Asymptotically optimal single-query sampling-based motion planning with quick replanning," *International Journal of Robotics Research*, 2015.

[111] J. Tu and S. X. Yang, "Genetic algorithm based path planning for a mobile robot," in *Proceedings of IEEE International Conference on Robotics and Automation*, pp. 1221 – 1226, 2003.

[112] G. Nagib and W. Gharieb, "Path planning for a mobile robot using genetic algorithms," in *International Conference on Electrical, Electronic and Computer Engineering. ICEEC '04*, pp. 185–189, Sept 2004.

[113] W. K. Chung and Y. Xu, "A generalized 3-d path planning method for robots using genetic algorithm with an adaptive evolution process," in *8th World Congress on Intelligent Control and Automation (WCICA)*, pp. 1354–1360, 2010.

[114] C. Zeng, Q. Zhang, and X. Wei, "Robotic global path-planning based modified genetic algorithm and A* algorithm," in *Proceedings of the Third International Conference on Measuring Technology and Mechatronics Automation*, vol. 03, pp. 167–170, 2011.

[115] A. Tuncer and M. Yildirim, "Dynamic path planning of mobile robots with improved genetic algorithm," *Computers & Electrical Engineering*, vol. 38, no. 6, pp. 1564–1572, 2012.

[116] L. Tian and C. Collins, "An effective robot trajectory planning method using a genetic algorithm," *Mechatronics*, vol. 14, no. 5, pp. 455–470, 2004.

[117] P. Shi and Y. Cui, "Dynamic path planning for mobile robot based on genetic algorithm in unknown environment," in *The 22nd China Control and Decision Conference*, pp. 4325 – 4329, 2010.

[118] M. Y. Ju and C. W. Cheng, "Smooth path planning using genetic algorithms," in *9th World Congress on Intelligent Control and Automation (WCICA)*, pp. 1103–1107, 2011.

[119] M. Cakir, "2d path planning of UAVs with genetic algorithm in a constrained environment," in *6th International Conference on Modeling, Simulation, and Applied Optimization (ICMSAO)*, 2015.

[120] J. D. S. Arantes, M. D. S. Arantes, C. F. M. Toledo, and B. C. Williams, "A multi-population genetic algorithm for UAV path re-planning under critical situation," in *IEEE 27th International Conference on Tools with Artificial Intelligence (ICTAI)*, 2015.

[121] W. Hornaday, C. Haas, J. O'Connor, and J. Wen, "Computer-aided planning for heavy lifts," *Journal of Construction Engineering and Management*, vol. 119, no. 3, pp. 498–515, 1993.

[122] K.-L. Lin and C. T. Haas, "An interactive planning environment for critical operations," *Journal of construction Engineering and Management*, vol. 122, no. 3, pp. 212–222, 1996.

[123] K. Varghese, P. Dharwadkar, J. Wolfhope, and J. T. O'Connor, "A heavy lift planning system for crane lifts," *Computer-Aided Civil and Infrastructure Engineering*, vol. 12, no. 1, pp. 31–42, 1997.

[124] S. Chadalavada and K. Varghese, "Development of a computer aided critical lift planning system using parametric modeling software," in *Proceedings of International Conference on Engineering, Project, and Production Management*, pp. 1–12, October 2010.

[125] H. Safouhi, M. Mouattamid, U. Hermann, and A. Hendi, "An algorithm for the calculation of feasible mobile crane position areas," *Automation in Construction*, vol. 20, no. 4, pp. 360–367, 2011.

[126] Z. Lei, H. Taghaddos, U. Hermann, and M. Al Hussein, "A methodology for mobile crane lift path checking in heavy industrial projects," *Automation in Construction*, vol. 31, pp. 41–53, 2013.

[127] Z. Lei, S. Han, A. Bouferguène, H. Taghaddos, U. Hermann, and M. Al-Hussein, "Algorithm for mobile crane walking path planning in congested industrial plants," *Journal of Construction Engineering and Management*, 2014.

[128] J. Olearczyk, M. Al-Hussein, and A. Bouferguène, "Evolution of the crane selection and on-site utilization process for modular construction multilifts," *Automation in Construction*, vol. 43, pp. 59–72, 2014.

[129] L.-C. Lien and M.-Y. Cheng, "Particle bee algorithm for tower crane layout with material quantity supply and demand optimization," *Automation in Construction*, vol. 45, pp. 25–32, 2014.

[130] P. Sivakumar, K. Varghese, and N. Babu, "Path planning of construction manipulators using genetic algorithms," in *IAARC/IFAC/IEEE. International symposium*, pp. 555–560, 1999.

[131] H. Reddy and K. Varghese, "Automated path planning for mobile crane lifts," *Computer Aided Civil and Infrastructure Engineering*, vol. 17, no. 6, pp. 439–448, 2002.

[132] J. Olearczyk, A. Boufergune, M. Al Hussein, and U. R. Hermann, "Automating motion trajectory of crane-lifted loads," *Automation in Construction*, vol. 45, pp. 178–186, 2014.

[133] S. Kang and E. Miranda, "Planning and visualization for automated robotic crane erection processes in construction," *Automation in Construction*, vol. 15, no. 4, pp. 398–414, 2006.

[134] Y. Lin, D. Wu, X. Wang, X. Wang, and S. Gao, "Lift path planning for a nonholonomic crawler crane," *Automation in Construction*, vol. 44, pp. 12–24, 2014.

[135] M. Gouttefarde, J.-F. Collard, N. Riehl, and C. Baradat, "Geometry selection of a redundantly actuated cable-suspended parallel robot," *IEEE Transactions on Robotics*, vol. 31, pp. 501–510, April 2015.

[136] J. Park, W.-K. Chung, and W. Moon, "Wire-suspended dynamical system: stability analysis by tension-closure," *IEEE Transactions on Robotics*, vol. 21, pp. 298–308, June 2005.

[137] S.-R. Oh and S. Agrawal, "Cable suspended planar robots with redundant cables: controllers with positive tensions," *IEEE Transactions on Robotics*, vol. 21, pp. 457–465, June 2005.

[138] M. Carricato and J. Merlet, "Stability analysis of underconstrained cable-driven parallel robots," *IEEE Transactions on Robotics*, vol. 29, pp. 288–296, Feb 2013.

[139] Y. Lin, D. Wu, X. Wang, X. Wang, and S. Gao, "Statics-based simulation approach for two-crane lift," *Journal of Construction Engineering and Management*, vol. 138, no. 10, pp. 1139–1149, 2012.

[140] H.-L. Chi and S.-C. Kang, "A physics-based simulation approach for cooperative erection activities," *Automation in Construction*, vol. 19, no. 6, pp. 750 – 761, 2010.

[141] P. Sivakumar, K. Varghese, and N. R. Babu, "Automated path planning of cooperative crane lifts using heuristic search," *Journal of Computing in Civil Engineering*, vol. 17, no. 3, pp. 197–207, 2003.

[142] M. Nowostawski and R. Poli, "Parallel genetic algorithm taxonomy," in *Knowledge-Based Intelligent Information Engineering Systems, 1999. Third International Conference*, pp. 88–92, IEEE, Dec 1999.

[143] Y. Tazaki and T. Suzuki, "Constraint-based prioritized trajectory planning for multibody systems," *IEEE Transactions on Robotics*, vol. 30, pp. 1227–1234, Oct 2014.

[144] Y. Cai, Z. Fan, H. Wan, S. Gao, B. Lu, and K. Lim, "Hardware-accelerated collision detection for 3d virtual reality gaming," *Simulation & Gaming*, vol. 37, no. 4, pp. 476–490, 2006.

[145] P. Cai, Y. Cai, I. Chandrasekaran, and J. Zheng, "A GPU-enabled parallel genetic algorithm for path planning of robotic operators," in *Y. Cai and Simon See (eds.), GPU Computing and Applications*, pp. 1–13, Springer, 2015.

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

[147] B. Gough, *GNU Scientific Library Reference Manual - Third Edition*. Network Theory Ltd., 3rd ed., 2009.

[148] E. Larsen, S. Gottschalk, M. Lin, and D. Manocha, "Fast distance queries with rectangular swept sphere volumes," in *IEEE International Conference on Robotics and Automation. Proceedings of ICRA '00*, vol. 4, pp. 3719–3726 vol.4, 2000.

[149] V. W. Tam and I. W. Fung, "Tower crane safety in the construction industry: A hong kong study," *Safety Science*, vol. 49, no. 2, pp. 208 – 215, 2011.

[150] R. A. King, *Analysis of crane and lifting accidents in North America from 2004 to 2010*. PhD thesis, Massachusetts Institute of Technology, 2012.

[151] C. Moenning and N. A. Dodgson, "Intrinsic point cloud simplification," *Proc. 14th GrahiCon*, vol. 14, p. 23, 2004.

[152] S. Gumhold, X. Wang, and R. MacLeod, "Feature extraction from point clouds," in *Proceedings of 10th international meshing roundtable*, 2001.

Appendix A

Kinematics

Most of "robots" considered in path planning are composed of rigid bodies linked together to form a *linkage* (Figure A.1(a)). In many cases the linkage can be simplified as kinematic chains (Figure A.1(b)) or kinematic trees (Figure A.1(c)). In a linkage, each rigid body is called a link A. two links A_1,A_2 in the kinematic tree are attached according to some motion constraints. The location where the two links are attached is referred to as *joints* as denoted as O in this Chapter. Studying the kinematics of a robot is beneficial for characterizing the C-space of the path planning problem. The C-spaces will be further discussed in Appendix B. It is also used for determine the position and motion ranges of the end-effector or other components of the robot which is required in collision detection. This section will start from the simplest kinematic chains then move to kinematic trees and introduce linkages with loops (Figure A.1(a)).

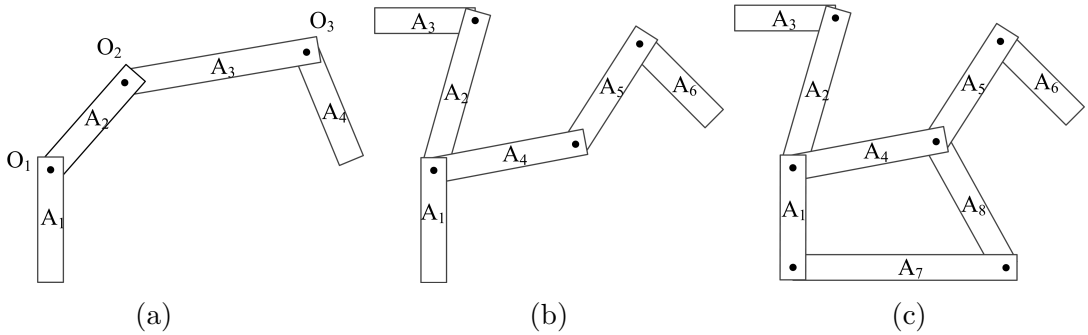


Figure A.1: Demonstration of linkages composed of rigid bodies attached by joints: (a) A linkage of rigid bodies with loops; (b) A kinematic chain of rigid bodies; (c) A kinematic tree of rigid bodies.

In R^2, a free rigid body possesses 3 independent parameters: x_t, y_t, θ. x_t and y_t stands for the amount of translation of the rigid body. θ represents the rotation of the rigid body along an anchor which can be either a randomly selected point or the joint position with another rigid body. For free rigid bodies in R^3, 6 independent parameters are required to characterize the motion: $x_t, y_t, z_t, \alpha, \beta, \gamma$. Here x_t, y_t, z_t characterize the translational motion of the rigid body in R^3 and α, β, γ can be the yaw-pitch-roll angles or the Euler angles or any variance of them. Quaternions may also be used to represent the rotational motion of the rigid bodies. The number of independent parameters to completely define the configuration of the rigid body is called the *Degree Of Freedom* (DOF) of the rigid body. The DOF is defined similarly for kinematic chains and trees.

Before considering the transformation of kinematic chains, the selection body coordinate systems of the rigid bodies in the kinematic chain need to be introduced. Given a 2D kinematic chain as in Figure A.1(b), the body frame of a link A_i is assigned such as the center is aligned to the joint O_{i-1} which links A_{i-1} and A_i and the x-axis is aligned to the direction from O_{i-1} to O_i. The center of the body frame of link A_1 can be placed randomly is since A_0 and O_0 are not defined. A denotation of the 2D body frame is shown in Figure A.2. The motion of A_i is constraint in only two ways in its body frame which leads to two types of 2D joints. When only rotation is allow in joint

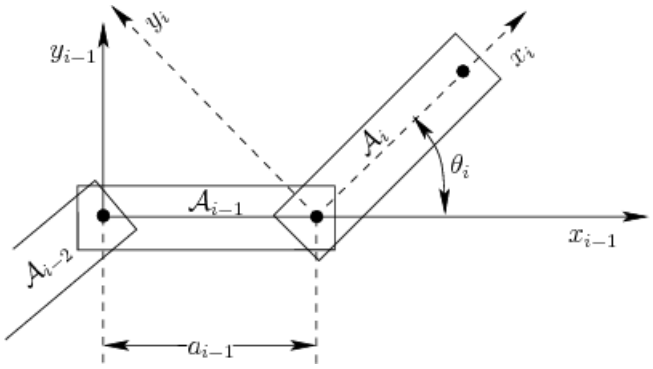


Figure A.2: Body frame of a link in R^2 (diagram courtesy of LaValle [3])

O_{i-1}, the joint is referred to as a *revolute* joint. Else if the only motion allowed is to move O_{i-1} along x_{i-1}, the joint is called a *prismatic* joint.

Joints for 3D kinematic chain have six major types. The *revolute* joints and *prismatic* joints are analogical to the 2D joints. A *screw* joint allows the link to rotation along a single axis. After each revolution, a uniform displacement along the rotational axis is conducted. A *cylindrical* joint allows to rotate and translate along a single axis. A *planar* joint enables a link to "slide" on the surface of another one. The last type which is the *sphere* joints allows to change all the rotational parameters. To select the body frames in a 3D kinematic chains, a common approach is to use the Denavit-Hartenberg (DH) parameters. To construct the DH parameters, the rotational axis of joint O_{i-1} is taken as the z axis z_i of link A_i. The direction perpendicular to both z_{i-1} and z_i is chosen as the x axis x_i. The y axis is selected as $y_i = z_i \times x_i$. The DH parameters include four values: d_i, θ_i, a_{i-1} and α_{i-1}. d_i and θ_i are the signed distance and angle from x_{i-1} to x_i measured along z_i. a_{i-1} and α_{i-1} represent the distance from z_{i-1} to z_i measured along x_{i-1}.

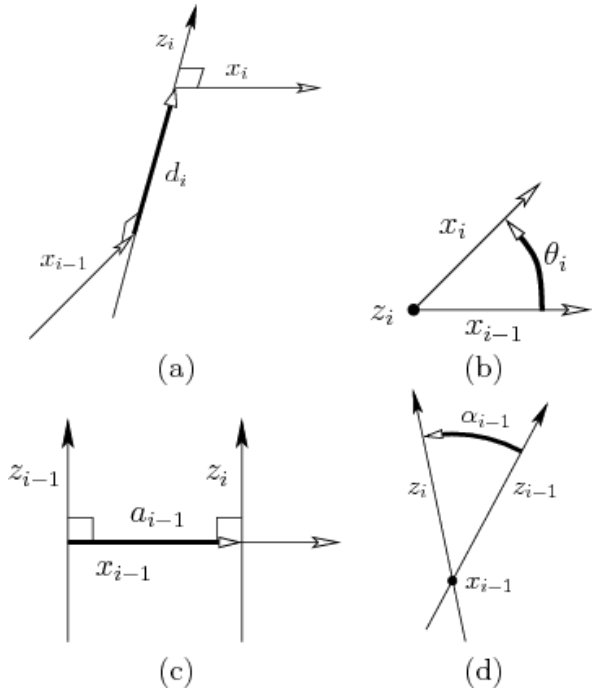


Figure A.3: The definition for the four DH parameters: (a) d_i; (b) θ_i; (c) a_{i-1}; (d) α_{i-1} (diagram courtesy of LaValle [3])

Each set of DH parameters for link A_i can be interpreted as a homogeneous transformation matrix represented as:

$$T_i = \begin{pmatrix} \cos(\theta_i) & -\sin(\theta_i) & 0 & a_{i-1} \\ \sin(\theta_i)\cos(\alpha_{i-1}) & \cos(\theta_i)\cos(\alpha_{i-1}) & -\sin(\alpha_{i-1}) & -\sin(\alpha_{i-1})d_i \\ \sin(\theta_i)\sin(\alpha_{i-1}) & \cos(\theta_i)\sin(\alpha_{i-1}) & \cos(\alpha_{i-1}) & \cos(\alpha_{i-1})d_i \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{A.1}$$

Given a kinematic chain $\{A_1, A_2, ..., A_n\}$, the world coordinates of the end effector at (x, y, z) in the body frame of A_i is computed as:

$$T_i = T_1 T_2 \dots T_n \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} \tag{A.2}$$

When a joint in the chain connects more than two links, the kinematic chain becomes a kinematic tree. The *root* of a kinematic tree is a link with only one joint attached. In order to determine the world coordinates of a point on link A_i, one need

first find a path connecting the root A_0 and A_i. For example, to determine the position of a point on A_6 in the kinematic tree shown in Figure A.1(b), the path A_1, A_4, A_5, A_6 shall be considered. The world coordinates of a point (x, y, z) in the body frame of A_6 can be calculated as:

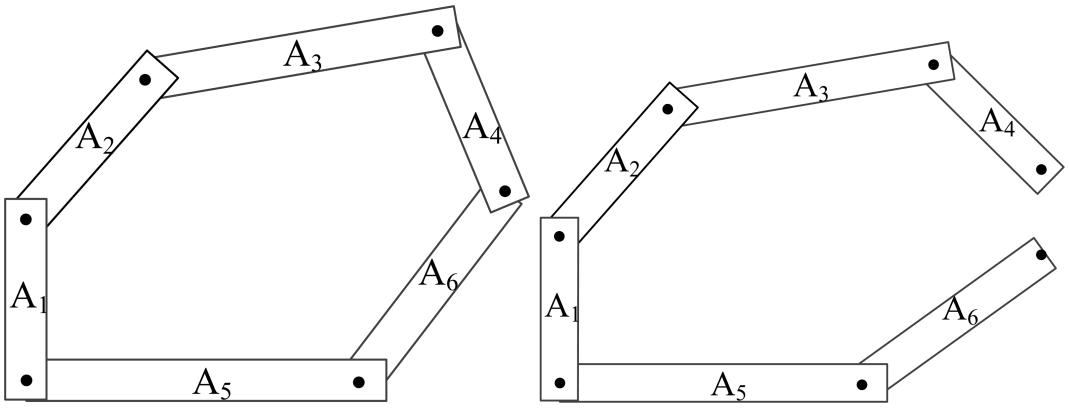
$$T_i = T_1 T_4 T_5 t_6 \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} \tag{A.3}$$

kinematic linkages with loops can be treated as kinematic trees through adding additional constraints. Consider a looped linkage shown in Figure A.4(a), a kinematic tree can be obtained through breaking the joint attaching link A_4 and link A_6. In order to make the kinematic tree a loop, the position of the joints at the end of the two branches have to be in the same position, which means that:

$$T_1 T_2 T_3 t_4 \begin{pmatrix} a \\ 0 \\ 1 \end{pmatrix} = T_5 T_6 \begin{pmatrix} b \\ 0 \\ 1 \end{pmatrix} \tag{A.4}$$

Where a and b stands for distance between the two joints in A_4 and A_6 accordingly.

This constraint contains two nonlinear equations (the last equation do not contain any variable) and therefore eliminates two degree of freedoms from the kinematic chain.



(a) (b)

Figure A.4: A looped linkage of rigid bodies: (a) The closed kinematic chain and (b) its corresponding open kinematic tree.

Appendix B

Configuration Space

Configuration space is the set of all possible configurations undertaken by a robot. Generally, the configuration space for a path planning problem with n DOFs is a n-dimensional manifold embedded in R^n. The shape of the C-space of robots differs according to their kinematic structure and operational limits. This section introduces the shapes of C-space of kinematic chains.

Consider a kinematic chain in which link A_1 is fixed on the x axis and A_2 is linked to A_1 through a revolute link. The only parameter to characterize this system is the angle θ_2 (or simply denoted as θ). The value of θ can be picked from a continuous range $[0, 2\pi)$ with 0 identified with 2π. Thus the C-space of this 2-body kinematic chain is homomorphic to the unit circle $\mathbb{S}^1$. If the links are attached through a prismatic joint instead, the parameter should be changed to x_2, the value of which lies between 0 and the length of A_1. In this case the C-space of the simple kinematic chain is $\mathbb{R}$. Furthermore, for a 2D kinematic chain $\{A_1, A_2, ..., A_{n+1}\}$ connected with revolute joints, the C-space is $T^n = \mathbb{S}^1 \times \mathbb{S}^1 ... \times \mathbb{S}^1$. If the first revolute joint is changed to prismatic joint, the C-space of the kinematic chain will become $R \times T^{n-1} = \mathbb{R} \times \mathbb{S}^1 ... \times \mathbb{S}^1$.

Table B.1: C-space contributed by the six types of joints in 3D kinematic chains

Joint type	**C-space**
Revolute joint	$\mathbb{S}^1$
Prismatic joint	$\mathbb{R}$
Screw joint	$\mathbb{R}$
Cylindrical joint	$\mathbb{R} \times \mathbb{S}^1$
Spherical joint	$\mathbb{SO}(3)$
Planar joint	$\mathbb{R}^2$

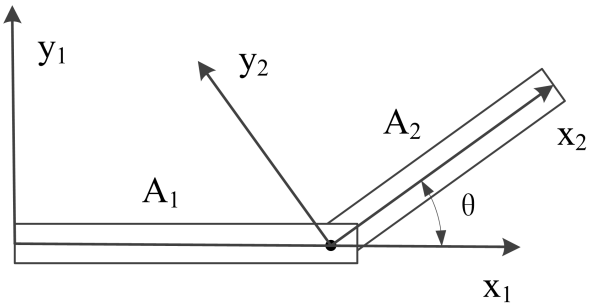


Figure B.1: A simple kinematic chain for C-space analysis

For 3D kinematics chains the representations are more complicates as there are more types of joints. Each type of joint contribute one or more dimensions to the overall C-space. Table B.1 lists the C-space contributed by the six types of joints in 3D kinematic chains. The overall C-space of the kinematic chain is the Cartesian product of the C-space contributed by each of the joints.