Real-Time Motion Planning and Simulation of Cranes in Construction

Homam Al-Bahnassi

A Thesis

in

The Department

of

Building, Civil and Environmental Engineering

Presented in Partial Fulfillment of the Requirements
for the Degree of Master of Applied Science (Building Engineering) at
Concordia University
Montreal, Quebec, Canada

January 2010

Your file Votre référence
ISBN: 978-0-494-67269-3
Our file Notre référence
ISBN: 978-0-494-67269-3

ABSTRACT

Real-Time Motion Planning and Simulation of Cranes in Construction

Homam Al-Bahnassi

Real-time planning the motion of heavy equipment (e.g. cranes) is an important issue in construction projects, where rapid and accurate planning directly affects the safety and productivity of operation. The work presented in this thesis is directed towards automatically generating an accurate motion plan in space and time for cranes by: (1) Investigating and utilizing motion planning algorithms to generate feasible paths with respect to all considered constraints; (2) Extending the efficiency of motion planning under complex global constraints (i.e. geometrical constraints) that represent static and dynamic obstacles found in the construction site; and (3) Considering local constraints that are related to the stability of the crane itself. Local constraints include engineering constraints (e.g. workloads for cranes) in addition to kinematic and dynamic constraints for the crane joints.

The methodology presented in this thesis was applied to develop a specialized motion planning system for construction equipment called Intelligent Construction Equipment motion Planner (ICE-Planner). This system was integrated into the 3D software to define, solve and visualize motion planning in real time.

The proposed methodology provides: (1) A motion planning framework for supporting cranes with the ability of generalizing over different types of equipment; (2) practical equipment planning which is aware of local constraints derived from engineering and kinematics properties of the equipment itself; (3) more accurate and realistic motion planning with efficiency in re-planning dynamic cases found in actual sites; and (4) the ability of visualizing and simulating motion planning results in real-time.

ACKNOWLEDGEMENTS

First and foremost, I wish to express my profound gratitude to my supervisor Dr. Amin Hammad for his continuous commitment, encouragement, and support. I feel privileged for having the opportunity to work with him and share passion for research.

I would like also to extend my appreciation to my committee members and to all professors who provided me with their valuable knowledge during my studies at Concordia and McGill.

I would like to pass my special thanks to my brother Wessam who supported me not only emotionally, but also with his help during the implementation of the prototype. Additionally, I would like to thank my colleague Chang Zhang for her kind support during my studies and in this research in addition to Mr. Jean-Louis Lapointe and Mr. Rafaele Palomar from GUAY crane company for giving us the opportunity to investigate cranes on site.

Finally I would like to sincerely thank the rest of my family, my parents, my sister and my fiancé -the latest family member-. Their love and support from the distance have greatly supported me and enabled all my accomplishments over the years.

TABLE OF CONTENTS

LIST OF FIGURES x

LIST OF TABLES xiv

ABBREVIATIONS xv

LIST OF MATHEMATICAL NOTATION xvii

CHAPTER 1 INTRODUCTION 1

1.1 GENERAL BACKGROUND 1

1.2 RESEARCH OBJECTIVES 3

1.3 THESIS STRUCUTRE 4

CHAPTER 2 LITERATURE REVIEW 5

2.1 INTRODUCTION 5

2.2 3D VISUALIZATION AND COMPUTER GRAPHICS 5

2.3 APPLICATIONS OF 3D VISUALIZATION IN CONSTRUCTION SIMULATION 8

2.4 ROBOTICS KINEMATICS 15

2.5 ROBOTIC SPATIAL REPRESENTATION 17

2.5.1 Configuration Space (C-space) 18

2.5.2 The Topology of C-space 19

2.5.3 Obstacles in C-space 20

2.6 ARTICIAL INTELLIGENCE 24

2.6.1 Agents 24

2.6.2 Path Planning Algorithms 26

2.6.3 Re-Planning Algorithms 36

2.6.4 Multi-Robot Planning Algorithms 39

2.7 CONSTRUCTION EQUIPMENT PATH PLANNING 41

2.8 ROBOTICS APPLICATIONS IN CONSTRUCTION AUTOMATION 45

2.9 SUMMARY AND CONCLUSIONS 48

CHAPTER 3 METHODOLOGY 49
3.1 INTRODUCTION 49
3.2 REAL-TIME MOTION PLANNING FRAMEWORK FOR CONSTRUCTION EQUIPMENT 50
3.3 KINEMATIC PROPERTIES OF A HYDRAULIC CRANE 58
3.3.1 C-space Dimensionality and Topology 64
3.4 ENGINEERING CONSTRAINTS 69
3.5 GEOMETRICAL REPRESENTATION 72
3.5.1 Increasing Safety by Avoiding Semi-Free Paths 74
3.5.2 Critical Volumes 76
3.6 THE SOLVER 78
3.6.1 Planning Algorithm 80
3.6.2 Enhancing the Algorithm 88
3.6.3 Multi-Equipment Planning/Re-Planning Algorithm 94
3.7 VISUALIZATION 99
3.8 SUMMARY AND CONCLUSIONS 101

CHAPTER 4 IMPLEMENTATION AND CASE STUDY 103
4.1 INTRODUCTION 103
4.2 SYSTEM COMPONENTS 104
4.3 SELECTION OF DEVELOPMENT TOOLS 105
4.4 WORKFLOW 108
4.4.1 Crane Definition 109
4.4.2 Engineering Constrains 116
4.4.3 Static Obstacles 118
4.4.4 Dynamic Obstacles 119
4.4.5 Defining Initial and Goal Configurations 121
4.4.6 Planning and Visualization 122
4.4.7 Re-Planning and Interactivity 127
4.5 IMPLEMENTATION DETAILS 128

4.5.1 Problem Modeling 131
4.5.2 Engineering Constraints 134
4.5.3 Planning/Re-Planning Solver 136
4.5.4 Visualization 137
4.6 VALIDATING AND EVALUATION 138
4.6.1 Description of the Case Studies 138
4.6.2 Validating the Results of the Case Studies 139
4.6.3 Evaluating the Results of the Case Studies 142
4.6.4 Effect of the Biasing Probability 146
4.6.5 Comparsion with the Basic RRT Algorithm 148
4.7 SUMMARY AND CONCLUSIONS 150

CHAPTER 5 SUMMARY, CONCLUSIONS AND FUTURE WORK 152
5.1 SUMMARY 152
5.2 CONCLUSIONS 154
5.3 LIMITATIONS AND FUTURE WORK 155

REFERNCES 159
APPENDICES 171
APPENDIX A: Programming-tree for the RRT visualization prototype created using the visual programming system in Softimage 171
APPENDIX B: Calculating tower crane homogenous transformation matrix based on DH-notation : (a) Schematic; (b) Homogenous transformation matrix; (c) Matlab code 173
APPENDIX C: Matlab code for calculating hydraulic crane homogenous transformation matrix symbolically based on DH-notation 174
APPENDIX D: C++ code for *Model3DRigidTreeXSI* Class that is used to model the crane kinematicaly and define the metric 175
APPENDIX E: C++ source code for the implemented *RRTBiasedConLim* variation 177
APPENDIX F: C++ source code for the implemented dynamic RRT algorithm 179

APPENDIX G: C# auto-generated source code for the engineering agent 181
APPENDIX H: Verbose logging for the solving process 184
APPENDIX I: Detailed results from the Scene Debugger 186
APPENDIX J: Reasons that cause capacity reduction for hydraulic cranes 188
APPENDIX K: List of Publications 190

LIST OF FIGURES

Figure 2.1: Steps for constructing 3D surface using the *Quickhull* algorithm (Ericson, 2005) 7

Figure 2.2: VITASCOPE animation snapshot of a construction site (Kamat and Martinez, 2001) 9

Figure 2.3: Automatic pouring system simulation (Zhou and Zhang, 2007) 10

Figure 2.4: The dual crane scenario (Chi et al., 2007) 11

Figure 2.5: Different approximation methods: (a) Spherical outer crane boundaries, (b) Cylindrical outer crane boundaries (Lai and Kang, 2009) 12

Figure 2.6: Crane positioned in a 3D environment (Cranimation, 2009) 14

Figure 2.7: Simlog training scenario (Simlog, 2009) 15

Figure 2.8: Parameters used in DH-notation (Craig, 2004) 16

Figure 2.9: The topology of the C-space for a two-joint planar arm: (a) two joint planner manipulator, (b) C-space representation of the manipulator as T^2 manifold (c) Unwrapped representation of the manifold (Adapted from Latombe, 2009) 20

Figure 2.10: The C-obstacle for a triangle robot with a fixed orientation (Adapted from LaValle, 2006) 22

Figure 2.11: Representation of obstacle B in C-space for a three DoFs (x,y,θ) robot A (Choest et al., 2005) 23

Figure 2.12: Obstacles representation in workspace and C-space: (a) Obstacles in the workspace of a robot; (b) The C-space showing the representation of the obstacles (Adapted from Choest et al., 2005) 24

Figure 2.13: Grid example of cell-decomposition method (Adapted from Bruce, 2009) 27

Figure 2.14: Working principle of potential field method: (a) Attractive Potential; (b) Repulsive Potential; (c) Sum of potentials (Dudek and Jenkin, 2000) 28

Figure 2.15: Example of a Visibility Graph (Fried et al., 2009) 29

Figure 2.16: Example of a Voronoi Diagram (Fried et al., 2009) 29

Figure 2.17: Steps of basic PRM algorithm (Adapted from Bruce, 2009) 30
Figure 2.18: Example of RRT with 2000 vertex (LaValle and Kuffner, 1999).....32
Figure 2.19: Growing two trees towards each other (Kuffner and LaValle, 2000).....33
Figure 2.20: Extended RRT with a waypoint cache for efficient re-planning (Bruce and Veloso, 2002) 38
Figure 2.21: DRRT steps for efficient re-planning (Ferguson et al., 2006).....39
Figure 2.22: Path of cooperative manipulators using GA search (Ali et al., 2005).....43
Figure 2.23: An example of the path refining process (top view) (Kang and Miranda, 2006).....44
Figure 2.24: Instrumented compactor (Bouvel et al., 2001).....46
Figure 2.25: Pavement inspection with obstacles: (a) Traversal Mode; (b) Longitudinal Mode; (c) Random Mode; (d) Grid Mode (Tseng et al., 2007) 48
Figure 3.1: Framework for construction equipment motion planning/re-planning 51
Figure 3.2: Flowchart for the proposed real-time motion planning framework: (a) Flow of the general framework; (b) Motion planning problem definition; (c) Planning/re-planning algorithm.....55
Figure 3.3: Defining the kinematic structure for a hydraulic crane: (a) Frames attached to the hydraulic crane; (b) Schematic for the hydraulic crane based on DH-notation 59
Figure 3.4: The effect of different hierarchy setups on the cable: (a) Cable linked directly to the boom; (b) Cable linked to intermediate frame that compensates the boom rotation.....62
Figure 3.5: Configuration constraint between boom rotation (θ_2) and cable rotation (θ_4) 64
Figure 3.6: Degrees of freedom for a crane 66
Figure 3.7: Transferring planning results from one prismatic joint to several joints 66
Figure 3.8: Example of transferring planning results from one prismatic joint to several joints 68
Figure 3.9: Working range (a) and load chart (b) of a crane (Groove Crane, 2006) 69
Figure 3.10: LVA pseudo code.....72

Figure 3.11: 3D tessellated model for a hydraulic crane 73
Figure 3.12: Applying different dilation values to a simple 2D obstacle (Adopted from Choset et al., 2005) 75
Figure 3.13: The path of one crane goes through critical volume that is bounded under the boom and the cable of the other crane: (a) perspective view; (b) top view 77
Figure 3.14: Points attached on the crane for defining a critical volume 77
Figure 3.15: The basic RRT algorithm (Adapted from LaValle, 1998) 81
Figure 3.16: The *Extend* function (Kuffner and LaValle, 2000) 81
Figure 3.17: RRT visualization for the first 3 DoF of a hydraulic crane: (a) Balanced RRT using the hypercube to calculate metric weights; (b) Unbalanced RRT using predefined weights for the metric 87
Figure 3.18: Including the goal-biasing into the *ChooseTarget()* function 88
Figure 3.19: The *Connect* function 89
Figure 3.20: The concept of the rules of action adapted *Connect* function 90
Figure 3.21: Adding rules of action modification to the *Connect* function 90
Figure 3.22: Pseudo code for the *Limited Greedy Connect* 91
Figure 3.23: The behavior of the *Limited Greedy Connect* function 92
Figure 3.24: *RRTBiasedLimCon* algorithm 93
Figure 3.25: Pseudo code for DRRT algorithm (Adapted from Ferguson, 2006) 97
Figure 3.26: A capture of tree that is trimmed and re-grown based on dynamic environment updates: (a) Initial planning; (b) Re-planning after detecting dynamic obstacle 98
Figure 3.27: Paths computed for a hydraulic crane and visualized in the work space 100
Figure 3.28: Two versions of the same crane model: (a) 3D model for the hydraulic crane, (b) Bounding-boxes for collision detection 101
Figure 4.1: Main system components and their relationships 104
Figure 4.2: RRT visualized in 3D space developed by ICE 108
Figure 4.3: Relationships between the computational and the visualization version 111

Figure 4.4: Plot of the function curves defined to drive the boom extension based on the computation results applied on the approximated model....112
Figure 4.5: Schematic view for the hierarchy of a hydraulic crane....113
Figure 4.6: Local kinematics property page used to define the DoF and its limits....115
Figure 4.7: Load chart of a hydraulic crane (Groove Crane, 2006)....116
Figure 4.8: Hydraulic crane engineering constraints GUI in Softimage....117
Figure 4.9: *MotionPlan_RealTimeDRRT* GUI....122
Figure 4.10: Plot of the RRT for the first DoF of the hydraulic crane....126
Figure 4.11: Excavator model moved near the crane and detected as dynamic obstacle while the crane is executing the path....128
Figure 4.12: Main components of the ICE-Planner....129
Figure 4.13: Sequence diagram of the ICE-Planner....130
Figure 4.14: I-beam dilated along its points normals....132
Figure 4.15: Simulation snapshots for the hydraulic crane case study....140
Figure 4.16: Simulation snapshots for the tower crane case study....141
Figure 4.17: Different feasible paths generated for the hydraulic crane case study....145
Figure 4.18: Screen capture for the Scene Debugger showing performance results....146
Figure 4.19: Relationship between the biasing probability and the proposed algorithm performance and results....147
Figure 4.20: Planning time of the two compared algorithms....149
Figure 4.21: Number of nodes in the tree of the two compared algorithms....149
Figure 4.22: Number of nodes on the path of the two compared algorithms....150
Figure A.1: Programming-tree for the RRT visualization prototype created using the visual programming system in Softimage....172
Figure J.1: Proper and improper cribbing of outrigger pads (Integrated Publishing, 2009)....189

LIST OF TABLES

Table 2.1: Summary of the comparison between different algorithms (Bruce, 2004) 35
Table 4.1: Values used in the evaluation tests 142
Table 4.2: Results summery for planning both case studies 142
Table 4.3: Results summery for re-planning both case studies 143
Table J.1: Sample table for capacity reduction caused by crane out of level (Integrated Publishing, 2009) 189

ABBREVIATIONS

2D	Two-dimensional
3D	Three-dimensional
4D	Four-dimensional
AI	Artificial Intelligence
ALS	Automated Landfill System
API	Application Programming Interface
BIM	Building Information Model
BBox	Bounding Box
CAD	Computer-Aided Design
CIC	Computer Integrated Construction
CIRC	Computer Integrated Road Construction
CMAG	Construction Metrology and Automation Group
CPU	Central Processing Unit
C-space	Configuration Space
DH-notation	Denavit-Hartenberg notation
DoF	Degree-Of-Freedom
DRRT	Dynamic Rapidly-exploring Random Trees
ERRT	Extended RRT
GA	Genetic Algorithms
GPS	Global Positioning System
GUI	Graphical User Interfaces
ICE-Planner	Intelligent Construction Equipment motion Planner
LVA	Local Validation Algorithm
MSL	Motion Strategy Library
NIST	National Institute of Standards and Technology
PQP	Proximity Query Package
PRM	Probabilistic Road Map
RRT	Rapidly-exploring Random Tree

SDK	Software Development Kit
Softimage ICE	Softimage Interactive Creative Environment
SRT	Scaling, Rotation and Translation
D*	Dynamic A*
BV	Bounding Volume
AEC	Architecture, Engineering, and Construction
UWB	Ultra Wideband

LIST OF MATHEMATICAL NOTATION

$\cup$	Union
$\cap$	Intersection
$\setminus$	Set difference
$\oplus$	Minkowski sum
$\in$	Element
C	Configuration space set
q	Configuration vector
C_{free}	Free space set
R	Real numbers
R^n	n-dimensioned real numbers
S^1	A circle
T^n	n-dimensional torus
S^n	n-dimensional sphere in R^{n+1}
$SO(n)$	Special Orthogonal group
$SE(n)$	Special Euclidean group
${}^{i-1}T_i$	Homogenous transformation matrix mapping frame i to frame $i-1$
$c\theta$	Cosine of the θ angle
$s\theta$	Sin of the θ angle
$\{i\}$	Coordinate system i
${}^{i-1}R_i$	Rotation matrix for frame i relative to frame $i-1$
${}^{i-1}P_i$	Position vector for frame i relative to frame $i-1$
T^T	Transpose of matrix T
T^1	Inversion of matrix T
$min(i-1,i)$	The minimum between value i and $i-1$
$max(i-1,i)$	The maximum between value i and $i-1$
$\Sigma(n,n-1,n-2,\ldots,n-i)$	Sum of a series of n number
$a\|b=c$	Given a, $b=c$

CHAPTER 1 INTRODUCTION

1.1 GENERAL BACKGROUND

Planning the motion of heavy equipment is an important issue in construction projects, where detailed and accurate planning directly affects the safety and productivity of operation. The increasing complexity of construction site conditions and equipments constraints makes manual planning either inefficient to prepare or inaccurate to execute. Taking cranes as an example of heavy construction equipment, from 1992 to 2006, there were 323 deaths related to cranes in the U.S. (NIOSH, 2009). These accidents were caused by contact with overhead power lines, workers struck by booms/jibs, struck by crane load, caught in between, etc. In Canada, there were 56 accidents related to cranes in the province of British Columbia in 2006 (WorkSafeBC, 2009); and during the period of 1974 to 2002, there were 23 accidents with injuries, 26 accidents with death, and 13 accidents with material damage related to cranes in Quebec province (CSST, 2009). Furthermore, the numbers of reported accidents and the resulting deaths are increasing during the past 10 years (Crane Accident Statistics, 2009).

Recent research tends to enhance safety and productivity by applying motion planning algorithms and simulation tools to plan and coordinate construction equipment. This still does not guarantee realistic and accurate motion planning for the equipment since it ignores completely or partially local constraints that represent the engineering, kinematic and dynamic constraints of the equipment itself, and focuses only on global constraints

caused by obstacles in the environment. Furthermore, current approaches represent obstacles as accurate, well-defined 3D models and ignore the uncertainty in their representation that is caused by the way capturing technologies work which results incomplete or low resolution 3D models that suffer from high uncertainty where a large number of small details are missing. Moreover, it is assumed that obstacles are static during the operation time ignoring the dynamic properties of the construction site. Finally, most implementations of the motion planning algorithms are customized towards specific types of equipment which limits the planning efficiency for realistic sites where different types of construction equipment can be found working together closely.

To fulfill tasks efficiently and safely in a complex environment with known and unknown obstacles, several methods are proposed for motion planning. During the planning stage, the model-based approach is used based on a 3D model of the site, which means initial information about the geometry of the equipment and the obstacles is given beforehand, so path planning becomes a one-time off-line operation. During the execution stage, the dynamic environment needs another approach, called sensor-based planning, with the assumption that some obstacles are unknown, and this is compensated by local on-line (near real-time) information coming from sensory feedback (Spong et al., 1992).

1.2 RESEARCH OBJECTIVES

The main objective of this research is to investigate real-time motion planning algorithms and simulation methods for planning heavy construction equipments (mainly cranes) while considering engineering constraints of equipments, in addition to the dynmic properties of construction sites. Several approaches are proposed to first represent a computational model of the equipment and the environment. Then planning algorithms are applied to find feasible paths that satisfy all equipment constraitns in dynamic construction environments. Our research objectives are:

1. To propose a framework for real-time motion planning that is applied for cranes as a case study and can be extended to other types of heavy construction equipment.
2. To investigate computational methods for modeling construction equipment and applying motion planning algorithms and 3D simulation tools to generate realistic planning.
3. To apply real-time intelligent decision-support system that is able to consider the dynamic properties of the construction environment by efficiently re-planning equipment paths.
4. To develop a prototype simulation system to test the above mentioned framework and algorithms in several case studies that can be the base for full construction equipment automation in risky environments.

1.3 THESIS STRUCTURE

In the next chapters, a literature review is discussed in Chapter 2 including 3D visualization and construction simulation, robotic kinematics and spatial representation, motion planning algorithms and agents. Chapter 3 proposes the research methodology, which includes a framework for real-time motion planning system, methods for deriving computational models of the problem, path planning and re-planning algorithms for construction equipment, and approaches for visualizing the results. In Chapter 4, requirements and the workflow for defining the motion planning problem, a summary of the implementation details, and the testing and validation are discussed. The research contributions and future work are described in Chapter 5.

CHAPTER 2 LITERATURE REVIEW

2.1 INTRODUCTION

This chapter focuses on related topics from construction simulation, robotics applications in construction equipment automation, motion planning for construction equipment and path planning algorithms. The purpose is to investigate the trends in research for utilizing such advanced algorithms and technologies from computer graphics, artificial intelligence and robotics to get real-time intelligent support for motion planning of construction equipment.

2.2 3D VISUALIZATION AND COMPUTER GRAPHICS

Computer graphics have become a mandatory tool for visualizing virtual worlds in which concepts, prototypes and simulations are developed. Such facility is capable of closely representing many real world aspects through 3D visualization by applying a series of operations on a virtual world stored as a digital 3D data set and representing it onto a display device (commonly a flat display panel).

There are various formats in which digitized 3D data can be expressed, each being more suitable to certain applications than others. Amongst these formats, polygonal surface modeling has become one of the most popular methods of 3D data representation, due to their flexibility and relative ease of use. As the name implies, models built with

polygonal modeling are made of a connected set of 3D polygons, forming a single continuous shell surface called a *polygon mesh*. The choice of the type of polygons in use is liberal, but most 3D rendering and collision detection libraries prefer using simple polygons such as triangles or quadrilaterals as the basic primitive type from which the entire mesh is made. Triangles are like atoms, in that the surface made of them can be rendered. The process of splitting polygons into more tractable primitives, such as triangles or quadrilaterals is called triangulation or, more generally, tessellation. Tessellation is the process of splitting a surface into a set of polygons (Akenine-Moller et al., 2008).

Polygonal meshes can be generated via several ways, including hand-modeling or surfacing of a cloud of 3D points. The latter method of generation relies on algorithms in the field of computational geometry. Two main approaches have been developed, the first is volume (or voxel) based approach, while the second approach is more boundary oriented (Amenta et al., 1996). Marching Cubes algorithm (Lorensen and Cline, 1987) is a good representative of the first approach. It constructs a high-resolution 3D surface by creating a polygonal surface representation of constant density from a 3D array of data. For the second approach, several algorithms are available that attempt to construct a polyhedral model of the object that bounds an unstructured cloud of points in 3D. One of these algorithms is the *Quickhull* Algorithm (Ericson, 2005). The concept for creating a 3D surface using this algorithm is illustrated for the 2D case in Figure 2.1. The first step shown in (a) locates the four extreme points (on the Bounding Box (BBox) of the point cloud), (b) all points inside the region formed by those points are deleted, as they cannot

be on the hull, (c) for each edge of the region, the point farthest away from the edge is located, (d) all points inside the triangular region so formed are deleted, and at this point the algorithm proceeds recursively by locating the points farthest from the edges of these triangle regions, and so on.

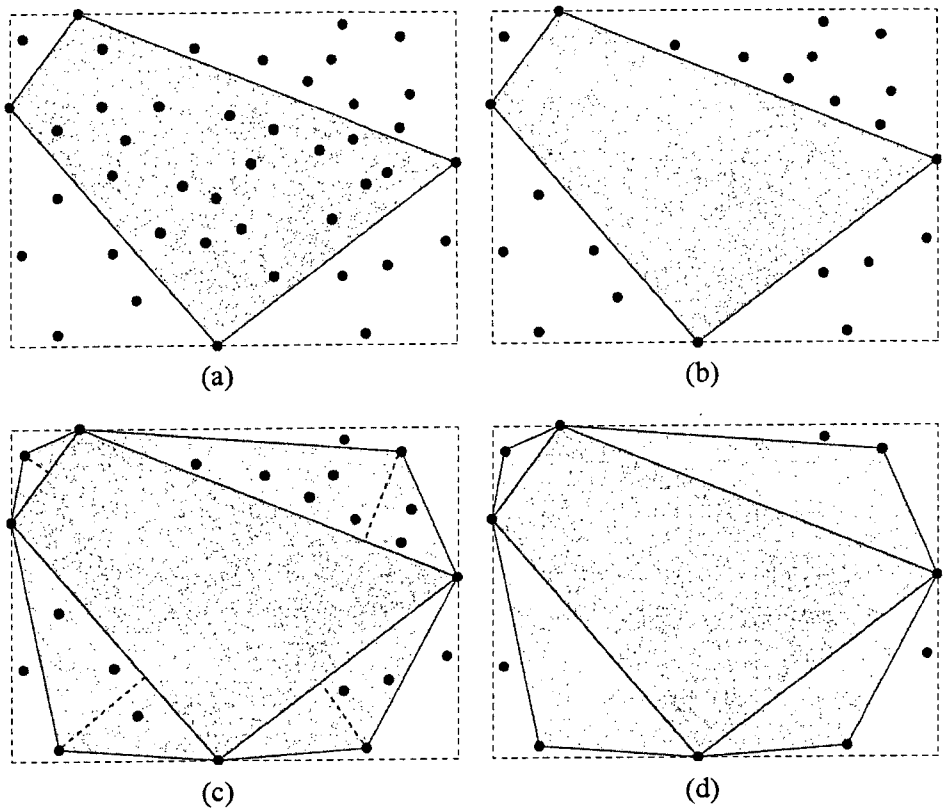


Figure 2.1: Steps for constructing 3D surface using the *Quickhull* algorithm (Ericson, 2005)

3D visualization can be accompanied with collision detection to provide reasoning about the virtual environment. Collision detection is part of what is often referred to as collision handling, which can be divided into three major parts: collision detection, collision

determination, and collision response. The result of collision detection is a Boolean saying whether two or more objects collide, while collision determination finds the actual intersections between objects; finally, collision response determines what actions need to be taken in response to the collision of the two objects (Akenine-Moller et al., 2008).

Directly testing the geometry of two objects for collision against each other is often very computationally expensive, especially when objects consist of hundreds or thousands of polygons. To minimize this cost, object bounding volumes are usually tested for overlap before the geometry intersection test is performed. A bounding volume (BV) is a single simple volume encapsulating one or more objects of more complex nature. The idea is for the simpler volumes (such as boxes and spheres) to have cheaper overlap tests than the complex objects they bound (Ericson, 2005).

2.3 APPLICATIONS OF 3D VISUALIZATION IN CONSTRUCTION SIMULATION

To achieve better understanding of construction processes, simulation tools have been developed to: (1) simulate and visualize these processes (Kamat and Martinez, 2001), (2) analyze and avoid collisions between equipment (Zhang et al., 2007), (3) test and visualize equipment location and manually plan the path (Cranimation, 2009; LiftPlanner, 2009), and (4) train operators of heavy equipment using virtual reality (Simlog, 2009). The advantage of visualizing the work processes is that the user can simulate and check the functional constraints and interferences that may happen in reality between the 3D physical elements and virtual workspaces.

Figure 2.2 shows an animation snapshot of a construction site. The visualization is based on the results of the simulation, which is not equipped with any collision detection mechanism, and it does not have any feedback about the unplanned environment changes. Therefore, if a spatial problem is detected in the visualization phase, the simulation has to be repeated after changing the input data.

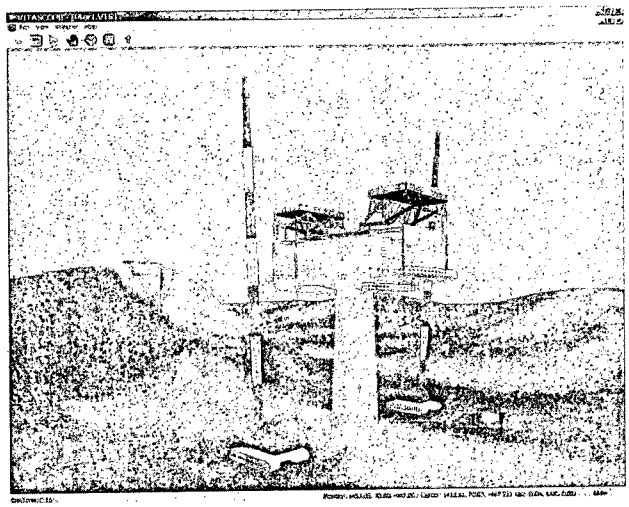


Figure 2.2: VITASCOPE animation snapshot of a construction site (Kamat and Martinez, 2001)

Figure 2.3 shows a simulation for automatic pouring system of a concrete boom pump. The trajectory of the boom is analyzed using inverse kinematics to define the path of the boom; however, no collision detection is applied between the boom sections and the obstacles in the environment.

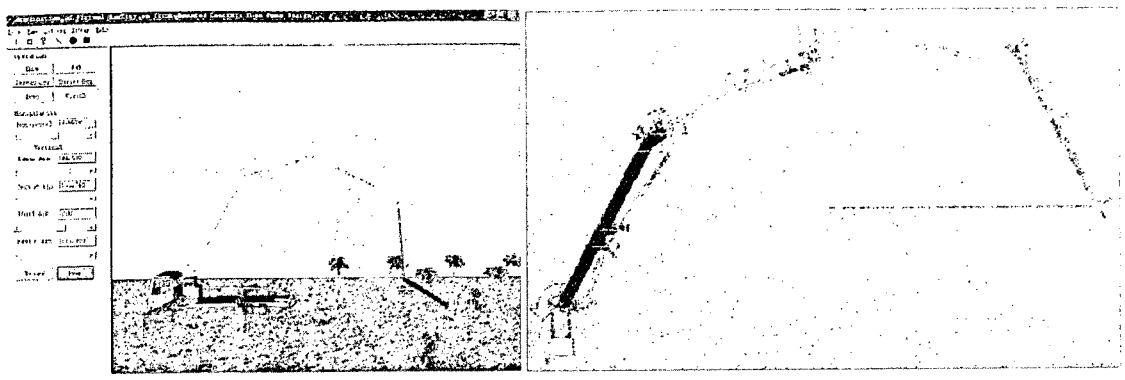


Figure 2.3: Automatic pouring system simulation (Zhou and Zhang, 2007)

A generic method to model and simulate construction cranes in operational details is discussed in the work of Kang (2006). The numerical model is used to help in visualizing crane activities on computers so that the goal of virtual construction environment can be realized. This work focused on rendering realistic animation of crane motions in a virtual environment where the generated animations are stored using the time histories of crane motions instead of the video formats to save disk space in long-period projects.

In the research of Chi et al. (2007), a prototype system is introduced for simulating and visualizing crane manipulation and cooperation. A dual-crane scenario is used to exemplify crane cooperation. Animation for crane usage is developed based on a numerical model that includes a suspension model for the cables and a manipulation model for the other components manipulated by the operator as shown in Figure 2.4. This prototype does not consider obstacles in the environment and only focuses on visualizing cranes operations.

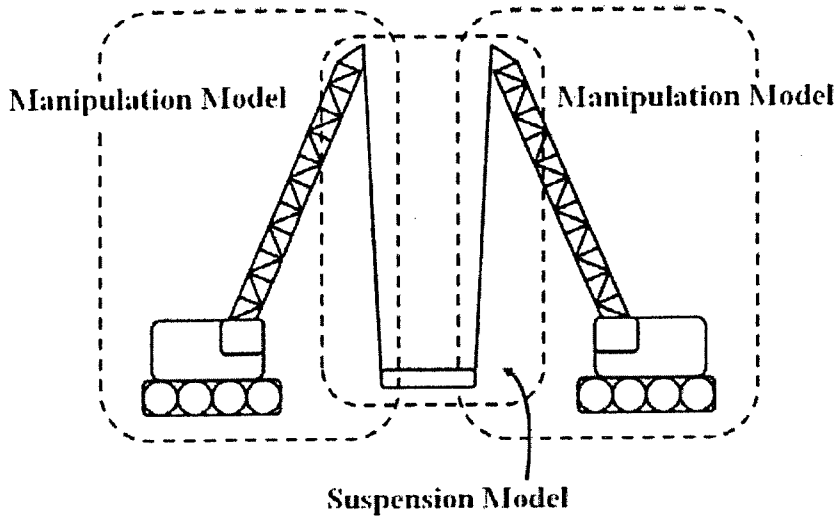


Figure 2.4: The dual crane scenario (Chi et al., 2007)

Lai and Kang (2009) presented simplified collision detection methods for visualizing a virtual construction scenario on a computer. In this work, different methods are used to approximate the common objects in construction sites, including tower cranes, mobile cranes, and structural elements. The approximation methods significantly reduce the computational cost by eliminating the collision checks between objects which are unlikely to collide. Although this work applies collision detection in construction simulation, it does not include any reaction events that affect the simulation based on the collision detection. Figure 2.5 shows different approximation methods applied to tower and hydraulic cranes.

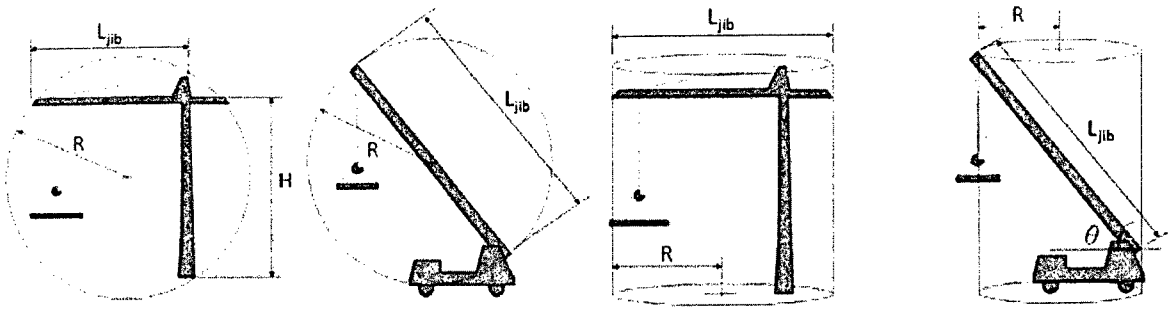


Figure 2.5: Different approximation methods: (a) Spherical outer crane boundaries; (b) Cylindrical outer crane boundaries (Lai and Kang, 2009)

A database system for automating the selection process of cranes on job sites was presented by Al-Hussein et al. (2000). This system includes operational information about crane geometry, lift configurations, lift capacity settings, accessories, and attachments. The developed database features, the capability of accommodating different types of cranes using different units of measurements, storing and querying capabilities and a user-friendly interface supported by graphics in a multimedia environment.

An optimization algorithm for selecting and locating mobile cranes on construction sites was presented by Al-Hussein et al. (2005). The developed algorithm and its optimization module provide practitioners with an evaluation tool for assessing lift configurations retrieved from the crane's database. It overcomes the limitations arising from the use of the limited information provided by the cranes' manufacturers in the form of lift-capacity charts.

Manrique et al. (2007) presented the concept used to construct a unique and complex tilt-up-panel structure utilizing an optimization model, 3D CAD, and animation. 3D animations were used to experiment with the construction process prior to construction in order to avoid potential costly on-site errors. In addition, the 3D animations were also used as a training tool for the contractors. This work focuses on describing the methodology used to integrate a crane selection algorithm and optimization model with 3D modeling and animation. The crane selection process followed the algorithm developed by Al-Hussein (1999); AutoCAD was used to develop the 3D objects of the crane, the panels and the site; MS-Solver was used for the optimization of the casting plate size and location of the cast panels, and to minimize the crane relocations; and 3D Studio MAX was used for the animation.

Cranimation (Cranimation, 2009) is a crane selection software, which calculates the outrigger forces for mobile cranes, the distribution of ground pressures for crawling cranes, and the minimum and maximum radius ranges. Figure 2.6 shows a sample result of positioning a crane on site. LiftPlanner (LiftPlanner, 2009) is a 3D crane and rigging planning software, which produces drawings to plan and document critical lifts. However, these software systems focus on the engineering constraints of the crane and provide detailed selection and configuration of the crane, but require the users to manually plan the path for moving the object considering obstacles in the 3D environment.

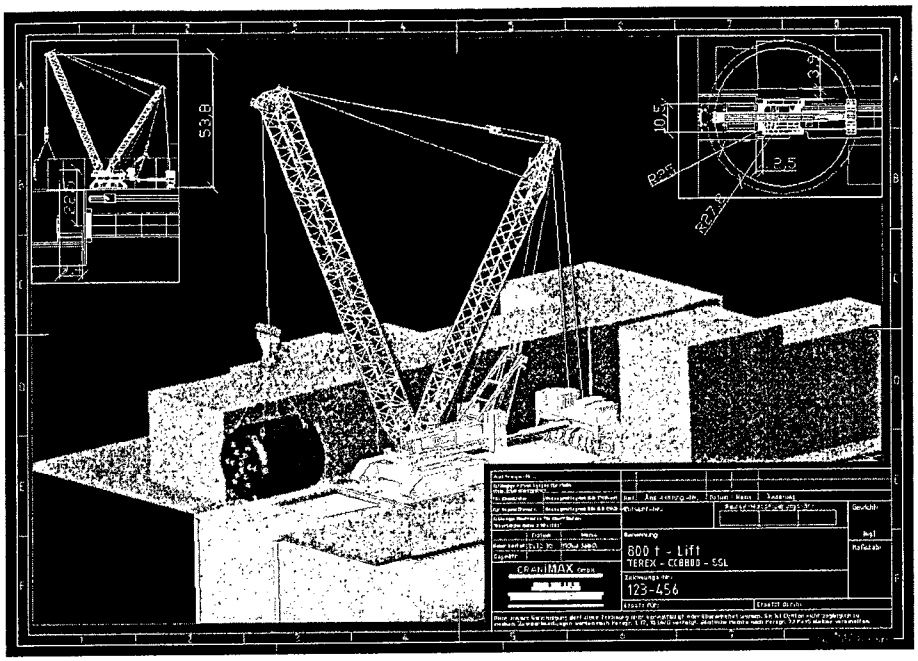


Figure 2.6: Crane positioned in a 3D environment (Cranimation, 2009)

Training simulation for equipment operation has been used as a cost-effective tool for the operators (Ritchie, 2004). Simlog (Simlog, 2009) provides training for different equipment with various scenarios, such as pouring concrete using bucket lifted by a tower crane (Figure 2.7). The shortcoming of the training software is that it does not have the real construction environment and it is not designed to provide path planning.

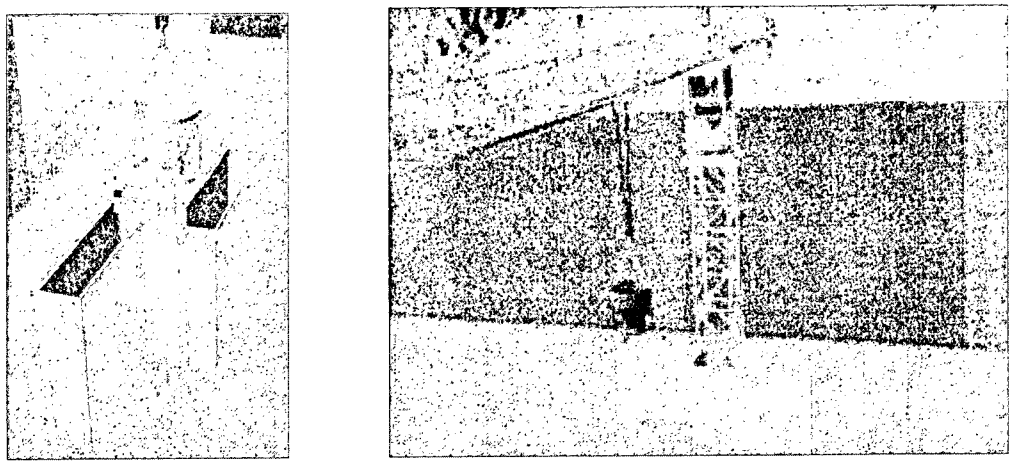


Figure 2.7: Simlog training scenario (Simlog, 2009)

The advantage of simulating construction environment and processes is providing a tool for visually checking potential collisions or other problems to ensure the reliability of a plan.

2.4 ROBOTIC KINEMATICS

Kinematics of a robot permit the study of its motion without regards to the forces which cause them. The purpose of the kinematic modeling of a robot is to describe its motion in a mathematical form. To kinematicaly describe a robot, firstly it is decomposed to a series of rigid links (rigid-bodies). These rigid links are connected by joints which allow relative motion of neighboring rigid bodies (links). Two types of joints are commonly used in robotics. The first one is the revolute joint which allows relative rotations between neighboring rigid links. The other one is sliding connections, also called prismatic joints, which allow relative displacement between links. The second step for defining the kinematics of a robot is to define the relationships between links.

This can be done based on the Denavit-Hartenberg notation (DH-notation) (Denavit and Hartenberg, 1955), where each link i is described relatively to its parent Link $i\text{-}1$ with four parameters, a_{i-1}, d_i, α_{i-1}, and θ_i. a_{i-1} denotes the length of Link $i\text{-}1$, which is the mutual perpendicular distance between Axis $i\text{-}1$ and Axis i. Likewise, a_i represents the length of Link i, which is the mutual perpendicular distance between the axes between Axis i and Axis $i+1$. d_i is called link offset. It is a signed distance measured along Axis i from the point where a_{i-1} intersects with the Axis i to the point where a_i intersects with Axis i. α_{i-1} denotes the twist angle between Axis $i\text{-}1$ and Axis i in the right-hand sense with respect to the direction of a_{i-1} from Axis $i\text{-}1$ to Axis i. θ_i is a parameter that describes the amount of rotation about this common axis between one link and its neighbor. Figure 2.8 shows these parameters.

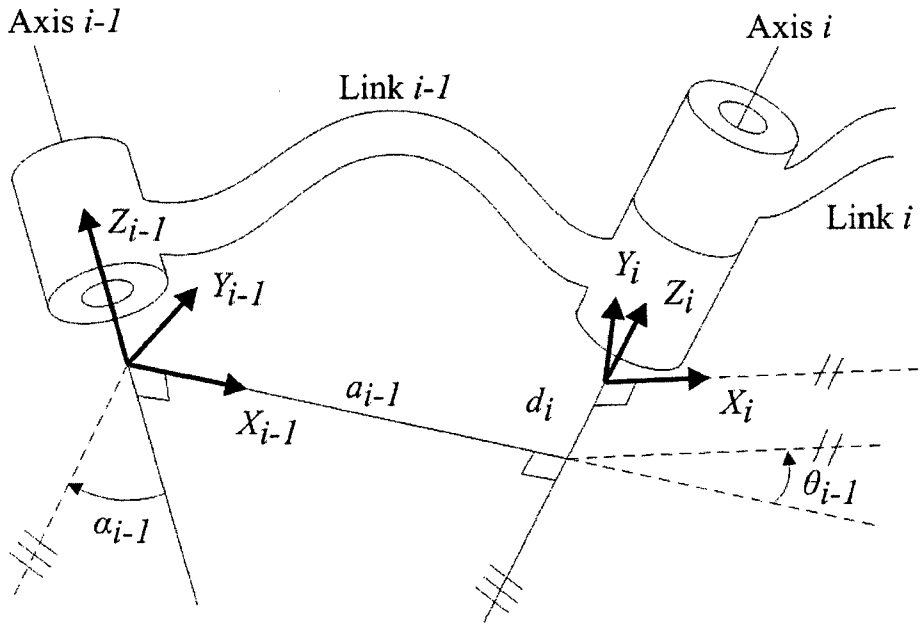


Figure 2.8: Parameters used in DH-notation (Craig, 2004)

By using the parameters defined in Figure 2.8, the relationship between Link *i-1* and Link *i* is essentially the transformation matrix between coordinate system $\{i\text{-}1\}$ and coordinate system $\{i\}$. This matrix can be represented in a homogeneous transformation matrix ${}^{i-1}T_i$ as follows:

$$ {}^{i-1}T_i = \begin{bmatrix} c\theta_i & -s\theta_i & 0 & a_{i-1} \\ s\theta_i c\alpha_{i-1} & c\theta_i c\alpha_{i-1} & -s\alpha_{i-1} & -s\alpha_{i-1}d_i \\ s\theta_i s\alpha_{i-1} & c\theta_i s\alpha_{i-1} & c\alpha_{i-1} & c\alpha_{i-1}d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad \text{(Equation 2.1)} $$

In order to obtain a more concise presentation, $c\theta_i$ is used to represent $cos(\theta_i)$, $s\theta_i$ is used to represent $sin(\theta_i)$, $s\alpha_{i-1}$ is used to represent $sin(\alpha_{i-1})$, and $c\alpha_{i-1}$ is used to represent $cos(\alpha_{i-1})$.

2.5 ROBOTIC SPATIAL REPRESENTATION

A robot's internal representation of its space is typically required for at least three different classes of tasks:

1. To establish which parts of the environment are free for navigation. This is a requirement to represent and manipulate the part of the environment that is free of obstacles. This region is known as free space.
2. To recognize regions or locations in the environment as special regions that are free but not safe.
3. To recognize specific objects within the environment (Dudek and Jenkin, 2000).

These requirements entail having a unified representation for the space in either the work space or a space representing the configuration of the robot, which is known as Configuration Space (C-space) (Choset et al., 2005).

2.5.1 Configuration Space (C-Space)

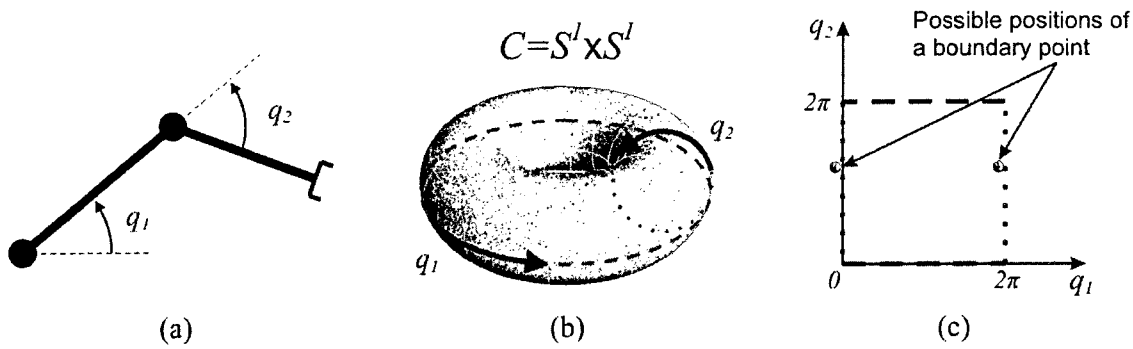
Most of the current approaches to path planning are based on the concept of C-space introduced by Lozano-Pérez and Wesley (1979). The C-space of the robot system is the space of all possible configurations of the system; and a configuration is simply a point in this abstract space. As a result, the dimensionality of the C-space is affected by the number of Degrees-of-Freedom (DoFs) of the robot. We should be able to specify the location of every point on the robot, since we need to ensure that no point on the robot collides with an obstacle. Once the C-space is generated, path planning requires only a search between the initial configuration (origin) and goal configuration (destination) in the C-space.

For cases where the robot has fewer controls than DoFs (C-space dimensionality is greater than controls) because of a variety of motion constraints that cannot be expressed as configuration constraints, it is considered a non-holonomic robot. A familiar example of such a system is a car. At low speeds, the rear wheels of the car roll freely in the direction they are pointing, but they prevent slipping motion in the perpendicular direction. This constraint implies that the car cannot translate directly to the side. This no-slip constraint is a non-holonomic constraint.

2.5.2 Topology of C-space

One reason that we care about the topology of C-space is that it will affect the representation of the space. Another reason is that if it is possible to derive a path-planning algorithm for one kind of topological space, then that algorithm may carry over to other spaces that are topologically equivalent.

In general, the C-space topology is different from the work space (W) topology. In some special cases, it can be the same. For example, in the case of a circular mobile robot, any $x_t, y_t \in R$ can be selected for translation, this alone yields an R^2 C-space which is the same as the work space. While for a two-joint planar arm, any rotation, $\theta \in [0, 2\pi)$, can be applied. But since 2π yields the same rotation as 0, they can be identified, which makes the set of 2D rotations into a manifold that wraps to a circle (S^1) for the first joint, and similarly for the second joint. To obtain the manifold that corresponds to all rigid-body motions, simply take $C = S^1 \times S^1$. The answer to the question is that the C-space is a kind of torus (T^2). Figure 2.9 shows the topology of the C-space for a two-joint planar arm: (a) two joint planner manipulator with two revolute DoFs (q_1 and q_2), (b) C-space representation of the manipulator as a manifold that wraps into both directions q_1 and q_2, and (c) Unwrapped representation of the manifold, note that it does not show the continuity of the DoFs. Thus for a boundary point, there will be two possible positions for it in the unwrapped representation.



(a) (b) (c)

Figure 2.9: The topology of the C-space for a two-joint planar arm: (a) two joint planner manipulator, (b) C-space representation of the manipulator as T^2 manifold (c) Unwrapped representation of the manifold (Adapted from Latombe, 2009)

If there are multiple bodies and they are allowed to move independently, then their C-spaces can be combined using Cartesian products. Let C_i denote the C-space of the rigid-body A_i. If there are n free-floating bodies in $W = R^2$ or $W = R^3$, then $C = C_1 \times C_2 \times ... \times C_n$. If the bodies are attached to form a kinematic chain or kinematic tree, then each C-space must be considered on a case-by-case basis. There is no general rule that simplifies the process (LaValle, 2006).

2.5.3 Obstacles in C-space

Previous subsections defined C-space in the absence of any collision constraints. The current subsection removes from C-space the configurations that either causes the equipment to collide with obstacles or cause some specified links of the equipment to collide with each other. The removed part of C-space is referred to as the obstacle region (C-obstacles). The leftover space is precisely what a solution path must traverse. A

motion planning algorithm must find a path in the leftover space from an initial configuration to a goal configuration.

We define a configuration space obstacle CB_i to be the set of configurations at which the robot intersects an obstacle WB_i in the workspace:

$$CB_i = \{q \in C | A(q) \cap WB_i \neq 0\} \quad \text{(Equation 2.2)}$$

where:

- q is the configuration
- C is the set of the C-space
- $A(q)$ is the robot set at configuration q in the workspace
- WB_i is the workspace obstacle

A C-obstacle region is defined as the union of all C-obstacles (CB_i): $\bigcup_{i=1}^{m} CB_i$. Based on the C-obstacle region, the free space can be defined by removing the C-obstacles region from C:

$$C_{free} = C \setminus \bigcup_{i=1}^{m} CB_i \quad \text{(Equation 2.3)}$$

To gain an understanding of how a C-obstacle can be constructed, consider the case of a triangular robot A and a rectangular obstacle B, robot A in the plane has three degrees of freedom, two for translation and one for rotation. For a fixed orientation, the

configuration space of the polygon is reduced to R^2 and the C-obstacle can be constructed by sliding the robot around the work space obstacle to find the constraints the obstacle places on the configuration of the robot, i.e., the possible locations of the robot's reference point as shown in Figure 2.10.

To include the orientation DoF of the robot, one way is to "stack" a set of two-dimensional configuration spaces, where each slice in the stack corresponds to the (x, y) configurations of the robot at a fixed orientation θ and the vertical axis represents the orientation of the robot. An example is shown in Figure 2.11.

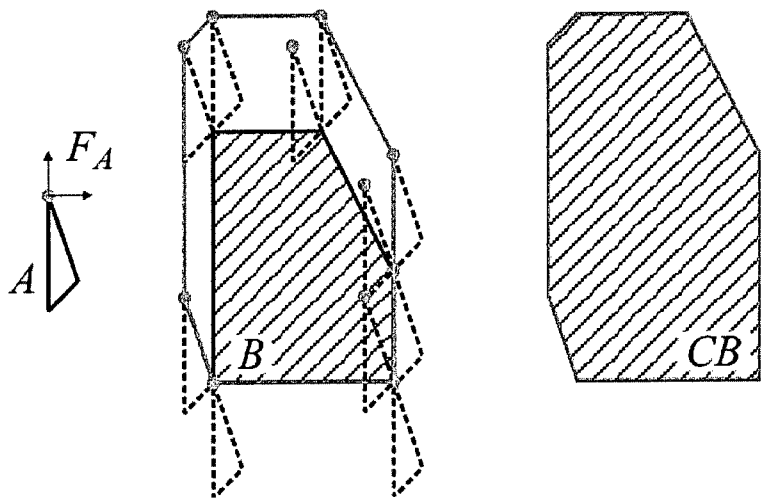


Figure 2.10: The C-obstacle for a triangle robot with a fixed orientation (Adapted from LaValle, 2006)

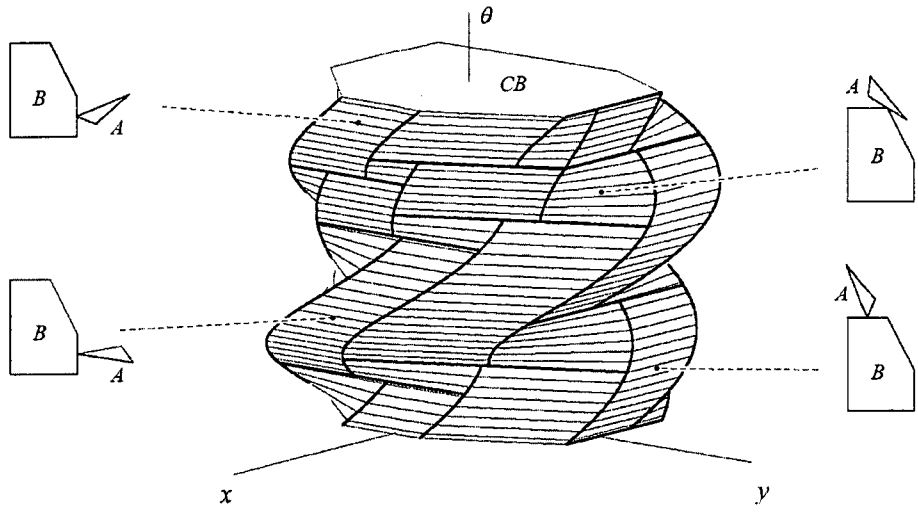


Figure 2.11: Representation of obstacle B in C-space for a three DoFs (x,y,θ) robot A (Choest et al., 2005)

Although both the workspace and the configuration space for this system can be represented by slices of R^2, and the obstacles appear to simply "grow" in each slice, the configuration space and workspace are different spaces, and the transformation from workspace obstacles to configuration space obstacles is not always so simple. For this reason, grid-based representations of the configuration space are sometimes used. For example for the case of a two-joint planar arm, for which C-space is T^2, a grid on the surface of the torus can be defined, and for each point on this grid a fairly simple test can be performed to see if the corresponding configuration causes a collision between the equipment and any obstacle in the workspace. If each grid point is represented by a pixel, the configuration space obstacle can be visualized by "coloring" pixels appropriately. Figure 2.12 shows how the C-obstacles (CB_1 and CB_2) for a 2 joint planner arm generated using this method.

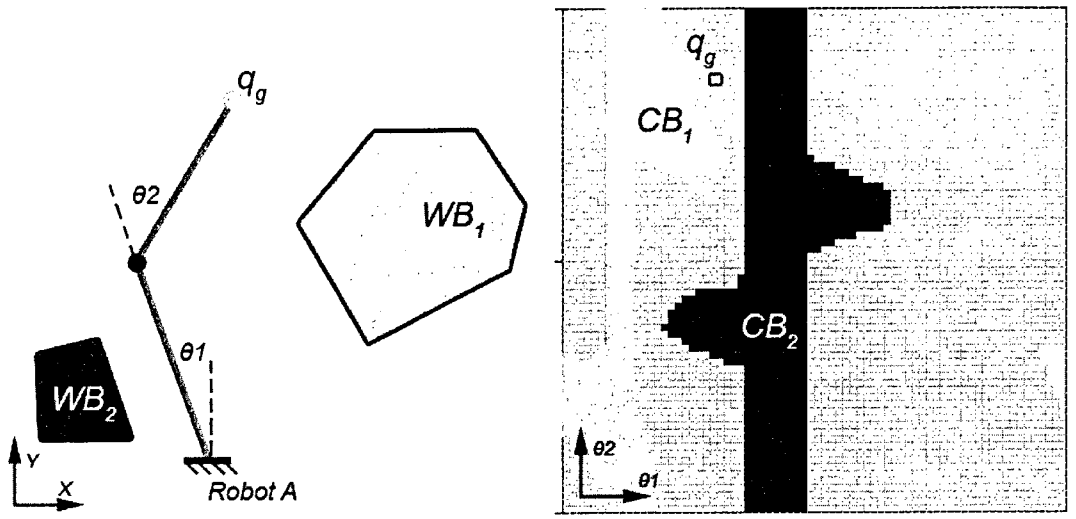


Figure 2.12: Obstacles representation in workspace and C-space: (a) Obstacles in the workspace of a robot; (b) The C-space showing the representation of the obstacles (Adapted from Choest et al., 2005)

The free configuration space (C_{free}) is the set of configurations at which the equipment does not intersect any obstacle in the C-space. In Figure 2.12 (b), the filled cells show the obstacle space, and the rest of the space is the free space for the robot.

2.6 ARTIFICIAL INTELLIGENCE

2.6.1 Agents

The concept of agents comes from developing a thinking machine with the capability of solving a problem on its own. An agent can be a computer program that is capable of accomplishing tasks on behalf of its user. Agents has provided the foundation for computers to deal with complex tasks, such as monitoring and controlling industrial

processes, assisting in medical diagnosis, or designing new machines. Agents are relatively independent and autonomous entities operating within communities in accordance with complex modes of cooperation, conflict and competition in order to survive and perpetuate themselves (Russell and Norvig, 2003).

Actions are taken by the agent to satisfy its objectives based on some satisfaction/survival function which it tries to optimise using its skills. The actions carried out by an agent will change the agents' environment, and thus its future decision making. Agents are endowed with autonomy, meaning that they are not directed by commands coming from a user, but by a set of tendencies, which can take the form of individual goals to be achieved or of satisfaction or survival functions which the agent attempts to optimise.

A number of researches involving agents have been done to utilize them in the construction industry for resolving problems. For example, agent systems have been used for construction claims negotiation (Ren and Anumba, 2002) and dynamic rescheduling negotiation between subcontractors (Kim and Paulson, 2003). Bilek and Hartmann (2003) have presented an agent-based approach to support complex design processes in Architecture, Engineering, and Construction (AEC). Zhang et al. (2009) has presented recent research on the application of agents together with Ultra Wide Band (UWB) technology in construction.

2.6.2 Path Planning Algorithms

Autonomous robotic path planning depends on path planning algorithms for generating collision-free path. A collision-free path from an initial configuration to a goal configuration of the equipment implies that at every step there is no collision between a piece of equipment and a static or dynamic obstacle in the environment.

Available algorithms for solving the path planning problem can be grouped under several categories: cell decomposition, roadmaps, potential functions and sampling based algorithms. Each algorithm inherits the advantages and disadvantages from the group it belongs to.

Cell-decomposition methods, including trapezoidal, quad tree or grid, depend on representing the free space by the union of simple regions, called cells, and encoding the adjacency relationships of the cells with an adjacency graph. Assuming the decomposition is computed, path planning with cell decomposition is about searching for a path within the adjacency graph. A* is common searching algorithm that is used to search for paths within adjacency graphs (Hart et al., 1968). These planners show good results in low-dimensional C-spaces, but for high-dimensional C-spaces become inefficient to implement and solve (Choset et al., 2005). Figure 2.13 shows an example of the grid-based method to represent the free space.

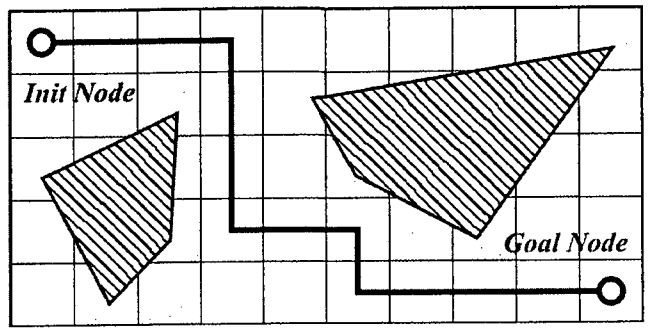


Figure 2.13: Grid example of cell-decomposition method (Adapted from Bruce, 2009)

Some of the most impressive results have been obtained using potential field methods. Such methods are attractive, since the heuristic function guiding the search for a path, the potential field, can easily be adapted to the specific problem to be solved, in particular the obstacles and the goal configuration. The main disadvantage of these planners is the presence of local minima in the potential fields. These minima may be difficult to escape. Local minima-free potential functions (also called navigation functions) have been defined in (Koditschek, 1987), (Rimon and Koditschek, 1992) and (Barraquand and Latombe, 1991). But these functions are expensive to compute in high-dimensional C-spaces and have not been used for many-DoF robots. Figure 2.14 shows the working principle of potential field method where: (a) the goal location generates an attractive potential – pulling the robot towards the goal, (b) the obstacles generate a repulsive potential – pushing the robot far away from the obstacles, (c) the sum result of the two potentials generated in (a) and (b).

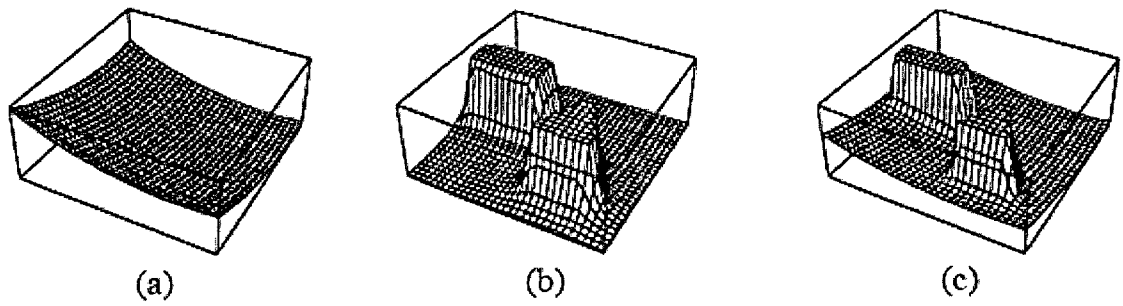


Figure 2.14: Working principle of potential field method: (a) Attractive potential; (b) Repulsive potential; (c) Sum of potentials (Adapted from Dudek and Jenkin, 2000)

Roadmap methods including the visibility graph (Lozano-Perez and Wesley, 1979), Voronoi diagram (O′ Dunlaing and Yap, 1982), and silhouette methods (Canny, 1988) have nice features that make them attractive for path-planning application in many fields. For example, the visibility graph has the advantage of optimality by generating the shortest path in a polygonal C-space, while Voronoi diagrams are good for safety since they are able to generate a safe path that has the maximum clearance around obstacles. All these three methods compute a roadmap that completely represents the connectivity of the free C-space. The visibility graph and Voronoi diagram methods are limited to low-dimensional C-spaces. The silhouette method applies to C-spaces of any dimension, but its complexity makes it less practical. Figure 2.15 shows a visibility graph that is formed by connecting all "visible" vertices, the start point and the end point, to each other. The thick line represents the shortest path between the initial and goal positions. Figure 2.16 shows Voronoi diagram formed by paths equidistant from the two closest objects.

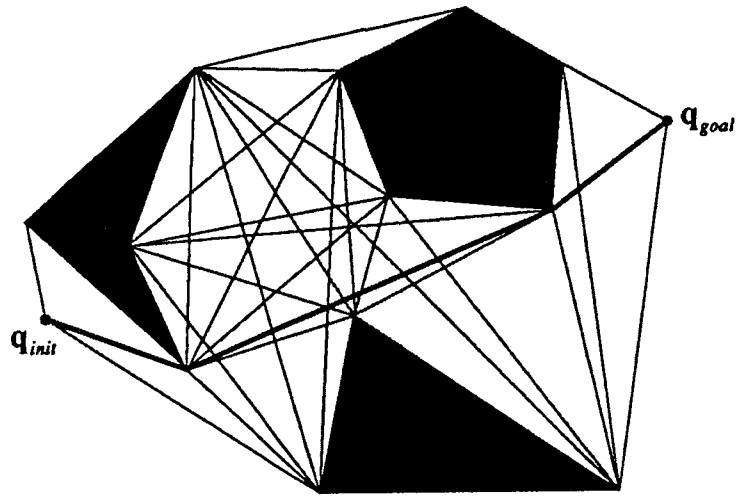


Figure 2.15: Example of a Visibility Graph (Fried et al., 2009)

Figure 2.16: Example of a Voronoi Diagram (Fried et al., 2009)

Considering sampling based algorithms, Probabilistic Road Map (PRM) planner samples the C-space for free configurations and tries to connect these configurations into a roadmap of feasible motions. There are a number of versions of PRM but they all use the same underlying concepts. The global idea of PRM is to construct a roadmap as a topological graph, and use it efficiently to solve multiple initial-goal queries. Intuitively, the paths on the roadmap should be easy to reach from each of initial and goal configurations (q_i and q_g), and the graph can be quickly searched for a solution (LaValle, 2006). The main advantage of PRM is that it is efficient for planning in high-dimensional C-spaces and for multi-query planning; but on the other hand, it suffers from environment updates because it needs to sample the C-space again to represent it correctly. Figure 2.17 shows the steps of the basic PRM algorithm: (a) Find random sample of free configurations (vertices); (b) Attempt to connect pairs of nearby vertices. If a valid local plan is found between two vertices, add an edge to the graph; (c) Find local connections to the graph from initial and goal positions; and (d) Search over roadmap graph.

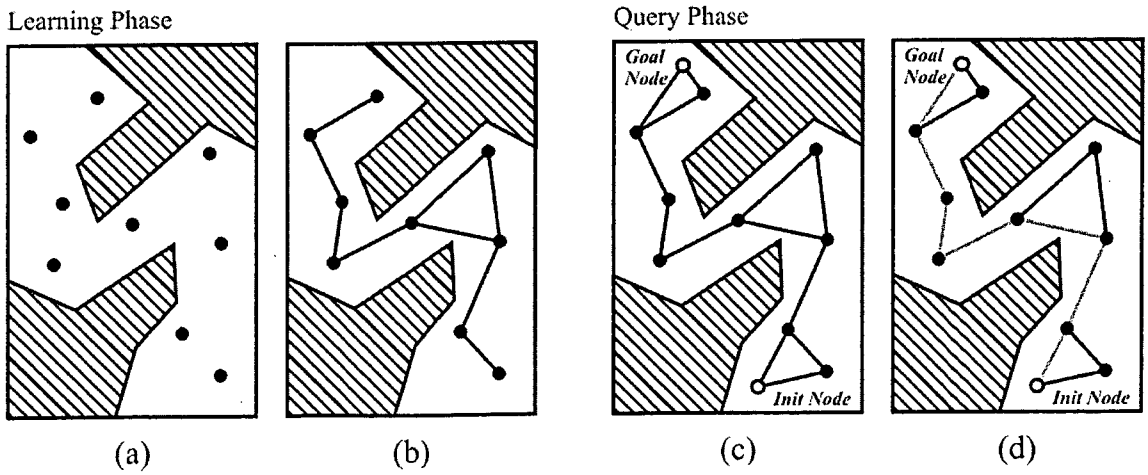


Figure 2.17: Steps of basic PRM algorithm (Adapted from Bruce, 2009)

Another important class of sampling based algorithms is the single-query sampling-based planners. These planners attempt to solve a query as fast as possible and do not focus on the exploration of the entire C_{free}. A representative for this class is the Rapidly-exploring Random Tree (RRT). RRT was introduced in (LaValle, 1998) as an efficient data structure and sampling scheme to quickly search high-dimensional spaces that have both algebraic constraints (arising from obstacles) and differential constraints (arising from non-holonomy and dynamics). These spaces are referred as state spaces where both types of constraints are considered. The key idea is to bias the exploration towards unexplored portions of the space. An RRT can be intuitively considered as a Monte-Carlo way of biasing search into largest Voronoi regions. Some variations can be considered as stochastic fractals. As with PRM, RRT is efficient in searching high-dimensional spaces and has many versions that can suite different planning problems. The main disadvantage of RRT is that it is not able to guarantee the generation of an optimal path based on pre-defined criteria; an optimization update is required to address this point. Fortunately, for many of these algorithms, the solutions produced are not too far from optimal in practice (LaValle, 2006). Brandt (2006) has made a comparison between A* and an RRT algorithm for motion planning of robots, and found that RRT is much faster than A*. Figure 2.18 shows an example of the RRT with 2000 vertex.

Other interesting lines of work include the method in (Kondo, 1991) which finds paths for 6-DoF manipulators using heuristic search techniques that limit the part of the C-space that is explored, and the planner in Ahuactzin et al. (1992) which utilizes genetic algorithms to search for a path in high-dimensional C-spaces.

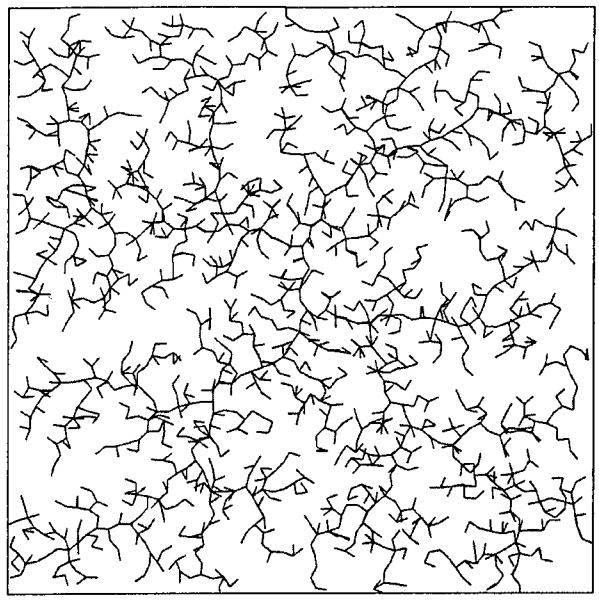


Figure 2.18: Example of RRT with 2000 vertex (LaValle and Kuffner, 1999)

A considerable share of the research happening in the field of motion planning focuses on introducing more efficient extensions to RRTs. Kuffner and LaValle (2000) introduced a simple and efficient extension to the RRT algorithm called *RRT-Connect* for solving single-query path planning problems in high-dimensional C-spaces. The method uses a simple greedy heuristic called "*Connect*" to extend the tree efficiently. It works by incrementally building two RRTs rooted at the start and the goal configurations. Each of the trees explores space around itself and also advances towards its other counterpart through the use of the greedy heuristic. Based on the comparisons between *RRT-Connect* and several other RRT-based variants, it is determined that for most problems, the greedy heuristic improves the running time, often by a factor of three or four, especially for uncluttered environments. In highly cluttered environments, the *Connect* heuristic only slightly increases running time in comparison to not using this heuristic in constructing

the two trees. Thus, it seems that the greedy behavior is worthwhile on average. Figure 2.19 shows an example of solving a simple planning problem using the *RRT-Connect*.

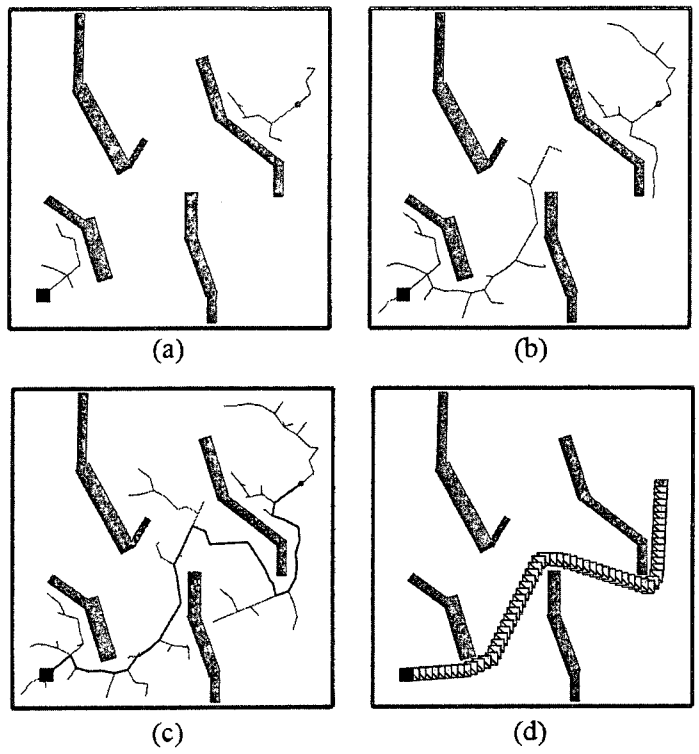


Figure 2.19: Growing two trees towards each other (Kuffner and LaValle, 2000)

A comparison of some of these algorithms is shown in Table 2.1, which is based on the following criteria:

1. Completeness: Complete planning approaches are guaranteed to find a solution when it exists, or correctly report failure if one does not exist (LaValle, 2006). For sampling based algorithms (e.g. RRT) completeness depends on the probability to find solution, the probability of them producing solution approaches 1 as more

time is spent. Improvements to the standard RRT can be done to address this issue (Cheng and LaValle, 2002). For Grid A*, finding the solution depends on the resolution of the grid that is representing the C space, low resolution grids may result in failure in finding the solution even if it exists. PRM combines both cases of being probabilistic and resolution complete, this is due to its nature of finding the path in two phases.

2. Optimal: In addition to completeness, algorithm optimality is considered as its ability to return optimal path with respect to some metric. For single-query sampling based algorithms (e.g. RRT), they are not able to guarantee the generation of an optimal path based on pre-defined criteria; an optimization update is required to address this point. Fortunately, for many of these algorithms, the solutions produced are not too far from optimal in practice (LaValle, 2006).
3. Efficient World Updates: Modifying the obstacles in the world is a very common case. Therefore, efficiency in re-planning the path after updating the world is important. In all the algorithms reviewed in this paper, there is no perfect algorithm for this type of query. RRT is the best even though it is considered semi-efficient in this type of queries.
4. Efficient Query Updates: In addition to world updates, query updates efficiency is important for cases like re-planning to new goals while fixing world constraints. The PRM algorithm is efficient in this type of queries, since it can reuse the roadmap that it constructed in the preprocessing phase.
5. Good DoF Scalability: The DoFs directly affect the complexity of C-spaces, thus configurations with high DoFs are not practical for solving by many algorithms.

Grid A* and visibility graph are not suitable for solving configurations with high DoFs.

6. Non-Holonomic: The capability of solving non-holonomic configurations is a key feature in path-planning algorithms, where the algorithm is not only limited of considering global constraints that are generated from explicit obstacles in the environment (Kuffner and LaValle, 2000), but it is also able to address local/differential constraints that may be found in some construction equipments. Among all reviewed algorithms, RRT stands with its high abilities in solving non-holonomic configurations.

Table 2.1: Summary of the comparison between different algorithms (Bruce, 2004)

Approach	Complete	Optimal	Efficient World Updates	Efficient Query Updates	Good DoF Scalability	Non-Holonomic
Grid A*	res	grid	no	no	no	no
Visibility Graph	yes	yes	no	no	no	no
RRT	prob	no	semi	semi	yes	yes
PRM	prob,res	graph	no	yes	yes	semi

Res: Resolution Complete, Prob: Probabilistic Completeness

2.6.3 Re-Planning Algorithms

In the case of motion planning for equipment on construction sites, every task has a schedule, and the unknown information can be assumed to be minor or less essential to the whole plan. Therefore, a motion re-planning approach can be efficient by modifying the off-line plan based on real-time sensed data.

During re-planning, the equipment must either wait for the new path to be computed or move in an uncertain direction; therefore, rapid re-planning is essential. An efficient re-planning algorithm should be able to plan optimal traverses in real-time by incrementally repairing paths to the equipment's state as new information is discovered. Re-planning algorithms focus on the repairs to significantly reduce the total time required for the initial path calculation and subsequent re-planning operations. Deterministic re-planning algorithms such as D* (Dynamic A*) (Stentz, 1994) efficiently repair previous planning solutions when changes occur in the environment. With basic D*, optimal path planning can be achieve in real-time by incrementally repairing paths to the robot's state as new information is discovered. However the disadvantage of the basic D* is that it propagates cost changes through all invalidated states without considering which expansions will benefit the robot at its current location. For improving the efficiency of re-planning with D*, like A*, heuristics can be used to focus the search in the direction of the robot and reduce the total number of state expansions. The work of Stentz (1995) propose an extension to the basic D* that utilize a heuristic function to focus the repairs based on the robot state. Thus, the computational costs are reduced by focusing the cost updates to minimize state expansions. This extension reduces the total time required for the initial

path calculation and subsequent re-planning operations. The net effect is a reduction in run-time by a factor of two to three. However, as the dimension of the search space increases, for example, in the case of multiple cranes working together, deterministic algorithms simply cannot cope with the size of the corresponding state space. On the other hand, randomized approaches such as RRTs are a good choice for solving this problem since they are not crippled by the high dimensionality. For re-planning with RRTs, Bruce and Veloso (2002) proposed an extension to the RRT algorithm called ERRT (Extended RRT) to build a real-time path planning system. This system is able to interleave planning and execution. It was evaluated first in a simulation environment, and then by applying it to physical robots. To improve re-planning efficiency and the quality of generated paths, they introduced two novel extensions to previous RRT work, the waypoint cache and adaptive cost penalty search. Based on the results, it is clear that ERRT is significantly more efficient for re-planning than the basic RRT planner, performing competitively with -or better than- existing heuristic and reactive real-time path planning approaches. A real robot was able to perform better than previous fully reactive schemes, traveling 40% faster while avoiding obstacles, and drastically reducing oscillation and local minima problems that the reactive scheme had. Figure 2.20 shows different cases for extending the RRT with waypoints cache.

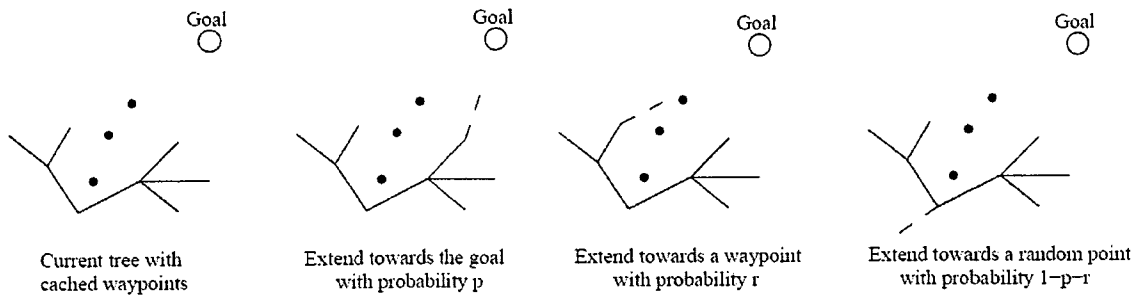


Figure 2.20: Extended RRT with a waypoint cache for efficient re-planning (Bruce and Veloso, 2002)

Another approach for re-planning is proposed by (Ferguson et al., 2006). In this approach, a re-planning algorithm is presented for repairing RRTs when new information concerning the configuration space is received. Instead of abandoning the current RRT entirely and re-growing a completely new RRT, this approach mimics deterministic re-planning algorithms by efficiently removing just the newly-invalid parts and maintaining the rest. It then repairs the tree be growing the remaining tree until a new solution is found. The resulting algorithm, called Dynamic Rapidly-exploring Random Trees (DRRT), is considered a probabilistic analog to the widely-used D* family of deterministic re-planning algorithms. The results demonstrate the usefulness of the DRRTs as good substitute for D* in high-dimensional problems. In comparison with ERRTs (as another alternative for re-planning with RRTs), on average, DRRTs outperformed ERRTs in each case by a factor of 5. Unfortunately, the experiments that are represented in work of Ferguson et al. (2006) are idealized and limited to a discretized limited environment. Utilizing the DRRT for solving problems with large continuous space is more challenging and requires further evaluation. Figure 2.21 shows the concept of DRRT for efficient re-planning: (a) an initial RRT generated from a start position to a

goal position, (b) a new obstacle is added to the C-space, (c) parts of the previous tree that are invalidated by the new obstacle are marked, (d) the tree is trimmed: invalid parts are removed, (e) the trimmed tree is grown until a new solution is generated.

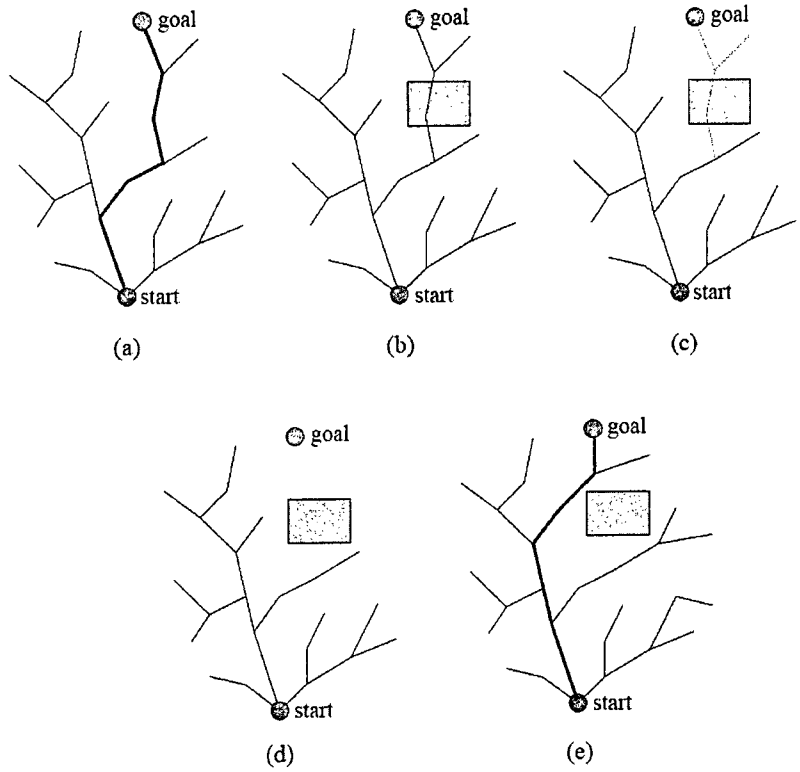


Figure 2.21: DRRT steps for efficient re-planning (Adapted from Ferguson et al., 2006)

2.6.4 Multi-Robot Planning Algorithms

The existing methods for solving the problem of motion planning for multiple robots can be divided into two categories (Latombe, 1991). In the centralized approach, the configuration spaces of the individual robots are combined into one composite configuration space which is then searched for a path for the entire composite system

(S´anchez and Latombe, 2002; Schwartz and Sharir, 1983). In contrast, the decoupled approach first computes separate paths for the individual robots and then resolves possible conflicts in the generated paths (LaValle and Hutchinson, 1998; Peng and Akella, 2003; Sim´eon et al., 2002). While centralized approaches (at least theoretically) are able to find the optimal solution to any planning problem for which a solution exists, their time complexity is exponential in the dimension of the composite C-space. In practice, one is therefore forced to use heuristics for the exploration of the huge joint state space.

Decoupled planners determine the paths of the individual robots independently and then employ different strategies to resolve possible conflicts. Accordingly, decoupled techniques are incomplete, i.e. they may fail to find a solution even if there is one. A popular decoupled approach is planning in the configuration time-space (Erdmann and Lozano-Perez, 1987) which can be constructed for each robot given the positions and orientations of all other robots at every point in time.

Another approach to decoupled planning is the path coordination method which was first introduced in O'Donnell and Lozano-Perez (1989). The key idea of this technique is to keep the robots on their individual paths and let the robots stop, move forward, or even move backward on their trajectories in order to avoid collisions (Bien and Lee, 1992).

A less studied approach to motion planning for multiple robots, which is nevertheless often used in practice, is prioritized planning (Bennewitz et al., 2002; Clark et al., 2002;

Erdmann and Lozano-Perez, 1987). In a prioritized approach, each of the robots is assigned a priority. Then, in order of decreasing priority, the robots are picked. For each picked robot a trajectory is planned, avoiding collisions with the static obstacles as well as the previously picked robots, which are considered as dynamic obstacles. This reduces the multi-robot motion planning problem to the problem of motion planning for a single robot in a known dynamic environment, which is a complex problem in itself, but not as complex as the centralized approach.

2.7 CONSTRUCTION EQUIPMENT PATH PLANNING

Although motion planning algorithms have been studied in computer science and robotics for more than thirty years, little research has been focusing on motion planning for construction equipment.

Hornaday et al. (1993) proposed a system for developing a computer-aided planning for heavy-lifts. The proposed system suggests the utilization of planning algorithms for fulfilling the requirements of the system, but it did not develop a suitable algorithm or investigates available path planning algorithms.

Tserng et al. (2000) proposed a methodology and several algorithms for interactive motion planning that are developed for multi-equipment landfill operations in an Automated Landfill System (ALS). It simulates the operational processes of landfill vehicles and equipment in pre-planning a landfill project as well as finding efficient and collision-free motion patterns to control autonomous landfill equipment during the

construction phase. However, this system depends on pre-defined patterns to do motion planning for the equipment. This prevents the system from solving actual cases where there could be equipment in the construction site that do not follow any of the specified patterns moving as dynamic obstacles.

Kim et al. (2003) introduced a path-planning method for a mobile construction robot to find a continuous collision-free path from the initial position of the construction robot to its goal position. This work presents an improved Bug-based algorithm, called SensBug, which can produce an effective path in an unknown environment with both stationary and moving obstacles. The contributions, which make it possible to generate an effective and short path, are: (1) an improved method to decide a local direction, which allows the mobile construction robot to generate an effective path in the environment with both stationary and movable obstacles; (2) a reverse mode, which can provide a mobile construction robot with a way to overcome the problem of obstacles in a complex configuration; and (3) a simple leaving condition, which allows the mobile construction robot to leave the obstacle boundary as soon as possible. These improvements make it possible to overcome the weak points of the previous algorithms. However, these improvements did not overcome the safety issue in all variations of bug algorithms where generated paths touch the obstacles. This issue is caused by the behavior of how bug algorithm navigates through the environment where it depends on wall-hugging the obstacles until it reaches the specified goal.

In the area of multi-equipment planning, Varghese and his colleagues have tried different algorithms, such as A* and Genetic Algorithms (GA) for optimizing the path for cooperative lift with two cranes (Sivakumar et al., 2003). In the research of Ali et al. (2005), a GA algorithm is used and compared with the A* algorithm, and the former is considered as a better solution for two cranes working together. Figure 2.22 shows a path traced by hook ends of two cooperative manipulators using a GA. However, they assumed that the site contains only static obstructions, and the proposed solutions only provide off-line planning, rather than real-time control of the movement. Gireesh and Vijayan (2007) have proposed a fuzzy logic approach to secure a collision free path avoiding multiple dynamic obstacles.

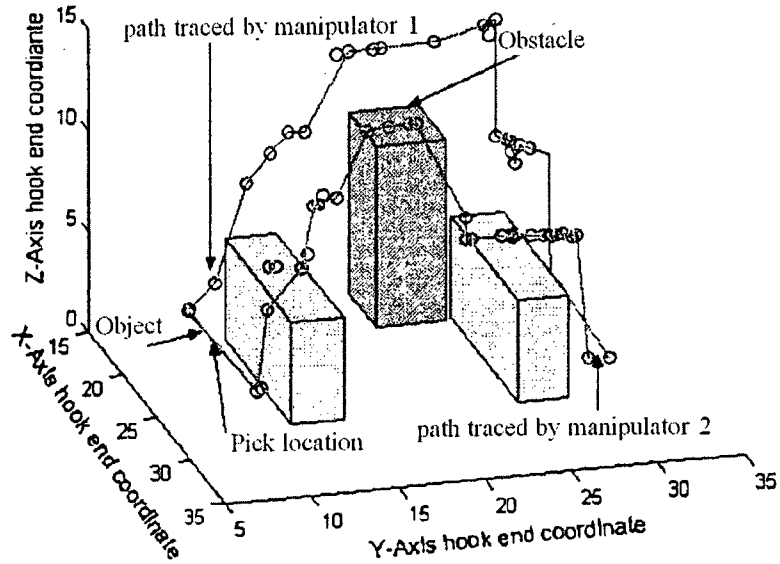


Figure 2.22: Path of cooperative manipulators using GA search (Ali et al., 2005)

Kang and Miranda (2006) have proposed an incremental decoupled method to plan motions for multiple cranes to avoid collision among any of the cranes as well as possible

collisions between the cranes and the transported objects. Three different algorithms, *QuickLink*, *QuickGuess*, and *RandomGuess*, were integrated to find a path efficiently. Moreover, path refining algorithms were developed to optimize a given path, as shown in Figure 2.23. Although this research considered dynamic changes on site to make the path more realistic, it assumed the environment information is known by following the work schedule.

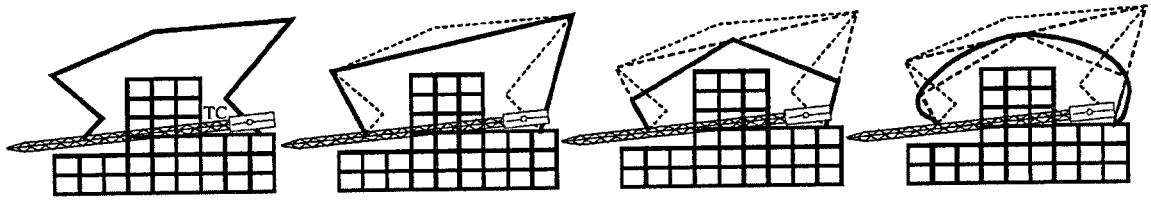


Figure 2.23: An example of the path refining process (top view) (Kang and Miranda, 2006)

Zhang et al. (2009) investigated motion planning algorithms to integrate them in collaborative multi-agent systems to support construction equipment operators using agents, wireless communication, and field data capturing technologies. Data collected from sensors attached to the equipment, in addition to an up-to-date 3D model of the construction site, are processed by the multi-agent system to detect any possible collisions or other conflicts related to the operations of the equipments, and to generate a new plan in real time. The potential advantages of the proposed approach are: more awareness of dynamic construction site conditions, a safer and more efficient work site, and a more reliable decision support based on good communications.

2.8 ROBOTICS APPLICATIONS IN CONSTRUCTION AUTOMATION

Unmanned construction is work performed by remotely operated construction machinery that corresponds to an operator controlled robot. Unmanned construction was used in civil engineering work for the first time in Japan in 1969 when an underwater bulldozer was used to excavate and move deposited soil during emergency restoration work at the Toyama Bridge that had been blocked by the Joganji River disaster. Since then, unmanned construction by excavators inside pneumatic caissons and by backhoes has been carried out. The restoration work following the volcanic eruptions that began in 1994 at the Unzen-fugendake Volcano and restoration work executed following the eruption of the Usuzan Volcano in 2000 were the first executions of large-scale unmanned construction and have spurred rapid progress in unmanned construction technologies and encouraged their wide use (Ban, 2002).

More research about construction automation is carried out in the National Institute of Standards and Technology (NIST, 2007) in the U.S. Construction Metrology and Automation Group (CMAG) is involved in the development of position/orientation tracking systems and sensor interface protocols. The Computer Integrated Construction (CIC) group is doing research on the visual representation and simulation of construction models (Furlani et al., 2002). Intelligent Systems Division with CMAG are researching robotic structural steel placement project called Automated Steel Construction Testbed (Lytle et al., 2002; 2004).

CMAG has been conducting research in crane automation since the mid 1980's. A robotic crane (RoboCrane) based on an inverted, cable actuated Stewart-Gough platform (2009) principle was invented at NIST at that time. Since then, several versions of the RoboCrane concept have been developed for various applications. Currently, CMAG is developing a generic crane controller using NIST real-time control system methodology in order to test and evaluate various automated crane control schemes. In addition, CMAG is working on methods and algorithms to identify construction components from high-resolution 3D laser scanning data and to determine their position and orientation. The use of low-resolution 3D range cameras for obstacle avoidance and crane load docking are also being investigated (Saidi and Lytle, 2008).

Computer Integrated Road Construction (CIRC) project has been aiming at introducing a new generation of control and monitoring tools for road pavement construction. Two prototypes are developed: CIRCOM for compactors (Bouvel et al., 2001), and CIRPAV for asphalt pavers (Peyret et al., 2000). Figure 2.24 shows a compactor instrumented with a GPS antenna, a gyro, a radar, and so on.

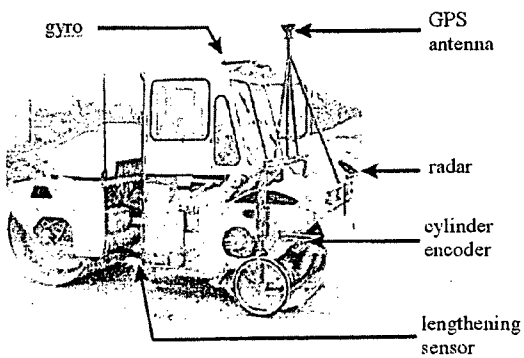


Figure 2.24: Instrumented compactor (Bouvel et al., 2001)

Oloufa et al. (2003) reports on research related to situational awareness of construction equipment using differential Global Positioning System (DGPS), wireless and web-based technologies. The technology presented here has a lot of promise, however, several areas have to be evaluated such as optimum system architecture, signal reliability, GPS accuracy and potential differential signal latency and communications issues.

The study of (Tseng et al., 2007) proposes the idea of using an autonomous robot to perform the pavement inspections for evenness and distress. Because inspection instruments are assumed to be mounted on the robot's platform, this robot is capable of inspecting pavement conditions while at the same time planning the upcoming motion path according to the inspection results. The authors propose four motion planning methods for robots as shown in Figure 2.25.

Unmanned and semi-automated construction systems could be used not only at disaster restoration sites, but also to increase safety and efficiency at ordinary construction sites. However, it is mentioned that the efficiency of unmanned construction is roughly 60% to 70% of that of manned construction, but sharply decreases in cases where the machinery moves or high precision work is necessary (Ban, 2002). Therefore, artificial intelligence (AI) methods can be used as an auxiliary tool to support the equipment operators, as was explained in Section 2.7.

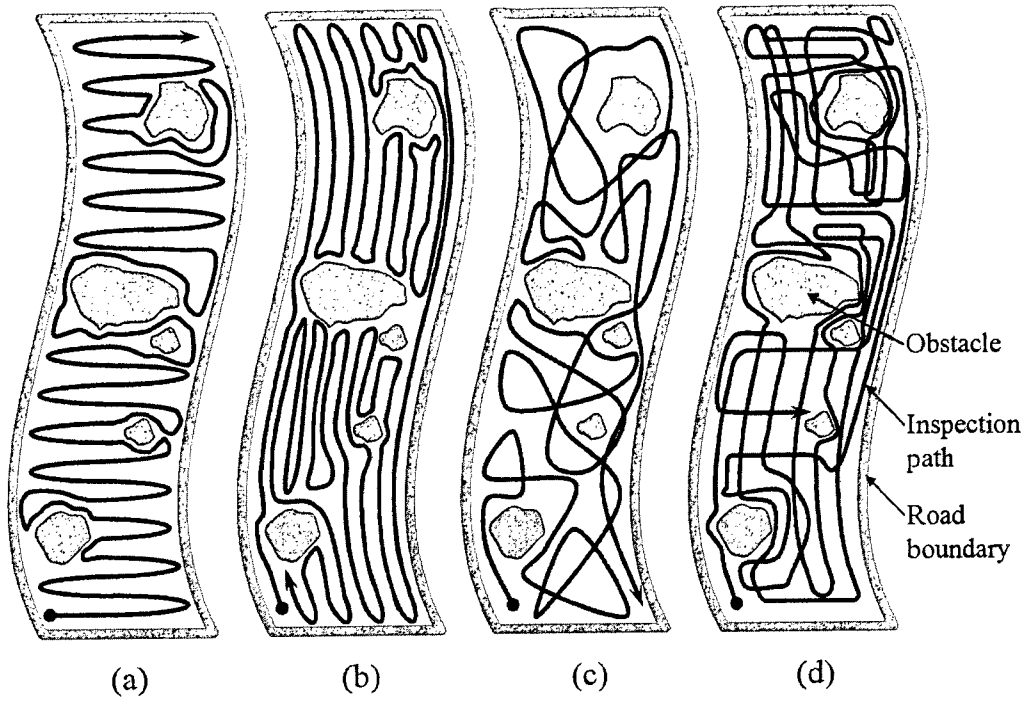


Figure 2.25: Pavement inspection with obstacles: (a) Traversal Mode; (b) Longitudinal Mode; (c) Random Mode; (d) Grid Mode (Tseng et al., 2007)

2.9 SUMMARY AND CONCLUSIONS

A wide range of literature from civil engineering, computer science, and robotics areas was reviewed, including automation in construction and current research trends. Different path planning algorithms were studied to select the most suitable one for real-time path re-planning in high-dimensional spaces that is found in construction equipments.

CHAPTER 3 METHODOLOGY

3.1 INTRODUCTION

As discussed in Chapter 2, the development of simulation software is enabling the visualization of equipment motion planning using virtual environments. However, many of these simulation tools focus on one side of the problem, and in all previous research approaches, the dynamic properties in the construction site are ignored. For example, some of the tools consider automatic motion planning, but make many simplifications and assumptions (e.g. ignoring engineering constraints). On the other hand, other tools consider engineering constraints, but they depend on manual motion planning. Therefore, an accurate motion planning system for coordinating the paths of construction equipment is necessary.

Current research shows trends towards integrating different disciplines including robotics, artificial intelligence and computer graphics to improve the operation of construction equipment. Since safety and productivity are major concerns when automating or supporting construction equipment operation, new approaches are required to generate realistic motion plans for construction equipment that can address the dynamic nature of the construction processes.

A computational model of the construction equipment can be built based on robotic concepts and computer graphics. Integrating this model into a 4D environment that represents the construction site with its dynamic characteristics generates a 4D simulation of the construction processes in near real-time. Based on the 4D simulation information, artificial intelligence algorithms can be applied to search for realistic motion paths under specified constraints to provide interactive intelligent support in near real-time. The results of the simulation are then visualized and analyzed in a 4D environment to help in decision-making before and while operating equipments.

In this chapter, a framework is developed to integrate and simulate real-time motion planning for construction equipment. The framework considers engineering constraints and the dynamic nature of construction sites. Several computational aspects of this framework are discussed, and it is used to develop a motion planning prototype system for construction equipment. The system is applied in case studies of hydraulic crane and tower crane as will be introduced in Section 4.6.

3.2 REAL-TIME MOTION PLANNING FRAMEWORK FOR CONSTRUCTION EQUIPMENT

The framework presented in this section shows the integration and relationships between the different components that are proposed for near real-time motion planning systems

for construction equipments. The general structure of the framework is shown in Figure 3.1.

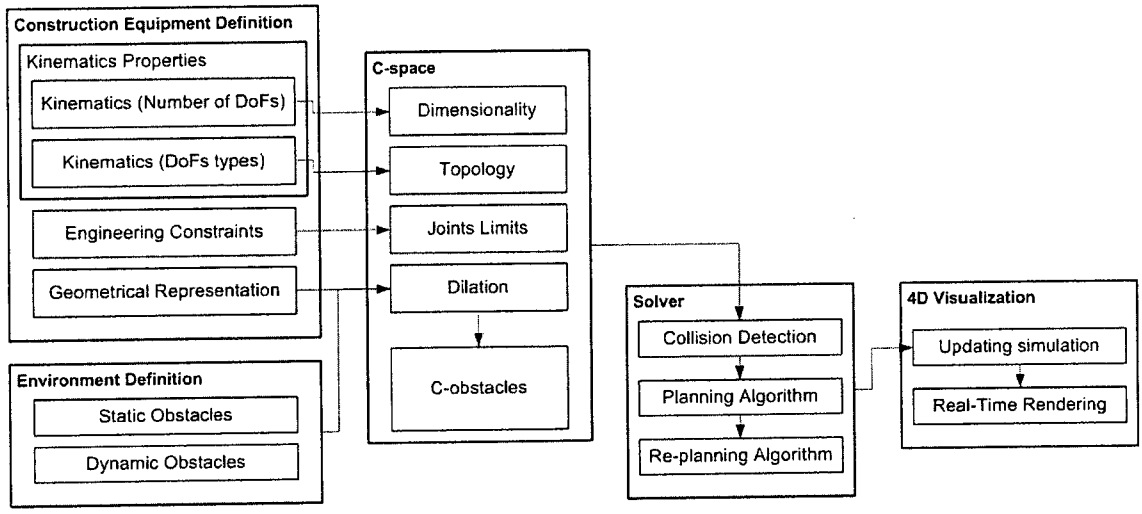


Figure 3.1: Framework for construction equipment motion planning/re-planning

The solver component in this framework is the core unit for computing feasible motion paths (planning), and modifying existing ones based on environment updates (re-planning). This component is based on algorithms that utilize continuous collision detection queries to search the C-space for optimized feasible paths. The C-space is generated based on the construction equipment and environment definitions. Under the Kinematics Properties, the number of DoFs of the equipment determines the dimensionality of the C-space, and the types of these DoFs specify its topology as described previously in Section 2.5.2. More modification to the C-space is done to accommodate the engineering constraints and dilation for ensuring realistic and safe

motion paths for the construction equipment. In the environment definition, the construction site environment is presented with either static or dynamic obstacles. Static obstacles represent all predefined known obstacles that are considered in the planning phase only. Dynamic obstacles represent all other obstacles that are not predefined prior to the planning phase and have high uncertainty that is reduced after executing the plan and only can be avoided with re-planning. Both of these two obstacle types are converted from their geometrical representation in the work space to C-obstacles representation in the C-space. The result of this conversion completes the generation of the C-space as it will be used by the solver to compute the optimized path.

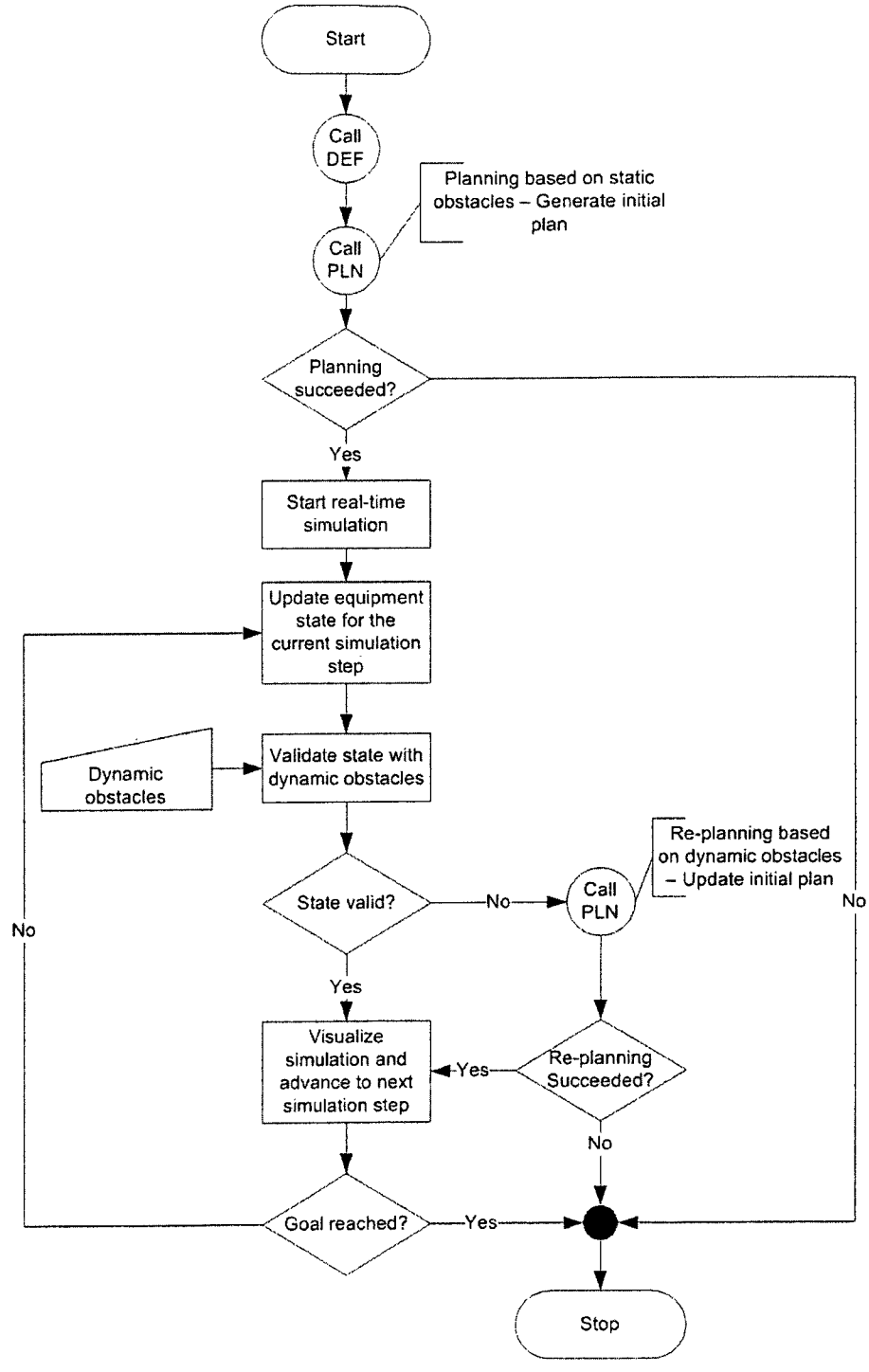
The visualization component in this framework is essential for viewing and analyzing the results in an interactive simulation environment. All results are rendered in a 4D simulation environment where users can easily navigate in the 3D space using navigation tools. It is necessary for the visualization to be able to render in real time since it should reflect the dynamic updates and the re-planning results as the solver modifies the motion paths. The high level of interactivity in the simulation environment enables the users to experiment with the dynamic nature of the construction site by interactively controlling the dynamic obstacles.

Figure 3.2 shows the flowchart of the proposed real-time motion planning framework. Figure 3.2 (a) starts with defining the motion planning problem (*DEF* sub-flowchart) that is presented in (b) and explained in detail in Section 3.3 and Section 3.5. The next step is planning the initial path based on the problem definition. The details for the planning

process are shown in *PLN* sub-flowchart (c) and its details are explained in Section 3.6. In case the planning process succeeded, the equipment starts updating its state based on the path for the first simulation step, if the planning failed to calculate a path, the simulation ends. The updated state is checked next against dynamic obstacles that are controlled interactively in the system. If the updated state is valid, it is rendered and the simulation advanced to the next step. Before repeating the process again for the next simulation step, the current step is checked if it is at the goal, where if that is true the simulation stops. In case of current state is not valid for dynamic obstacles, a re-planning process is executed and again it is checked for success, where it is either succeeded and visualized or failed and the simulation is stopped.

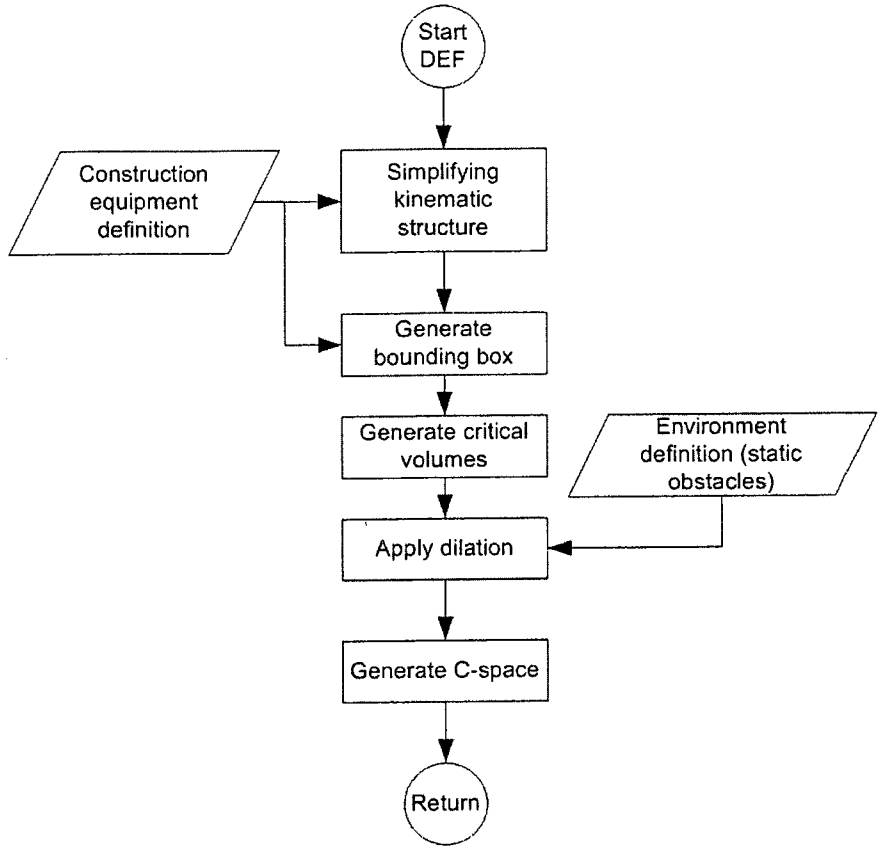
Figure 3.2 (b) shows an overview for the process of defining the motion planning problem in the proposed framework. It starts with two simplification processes for the construction equipment definition. One is for simplifying the kinematic structure explained in Section 3.3.1, while the other one is for simplifying the geometrical representation to bounding boxes and is explained in Section 3.7. The next two processes are responsible for increasing the safety of the generated path as described in Section 3.5. The first process generates critical volumes for preventing the solver from generating paths in unsafe volumes. The other process expands the obstacles defined in the environment to avoid unsafe paths that are too close to obstacles. This completes the definition of the motion problem as the C-space that can be searched by the planning/re-planning algorithms is generated.

Figure 3.2 (c) represents the general flow of the planning/re-planning algorithm developed in this research. This algorithm depends on RRTs as a randomized sampling planning algorithm. It starts first by sampling a random node in the C-space and growing a tree towards the sampled node. Details about the proposed approaches for sampling nodes and growing the tree are explained in Section 3.6.2. The resulted node of growing this tree is then validated in two phases; first phase is validating the node for obstacles then the second phase for engineering constraints. If any of these two validations failed, a new node is sampled and the process is repeated again. In the case of success, the node is added to the tree and the process is repeated again after checking if the grown tree has reached the goal. Details about the developed algorithm are explained in Section 3.6.



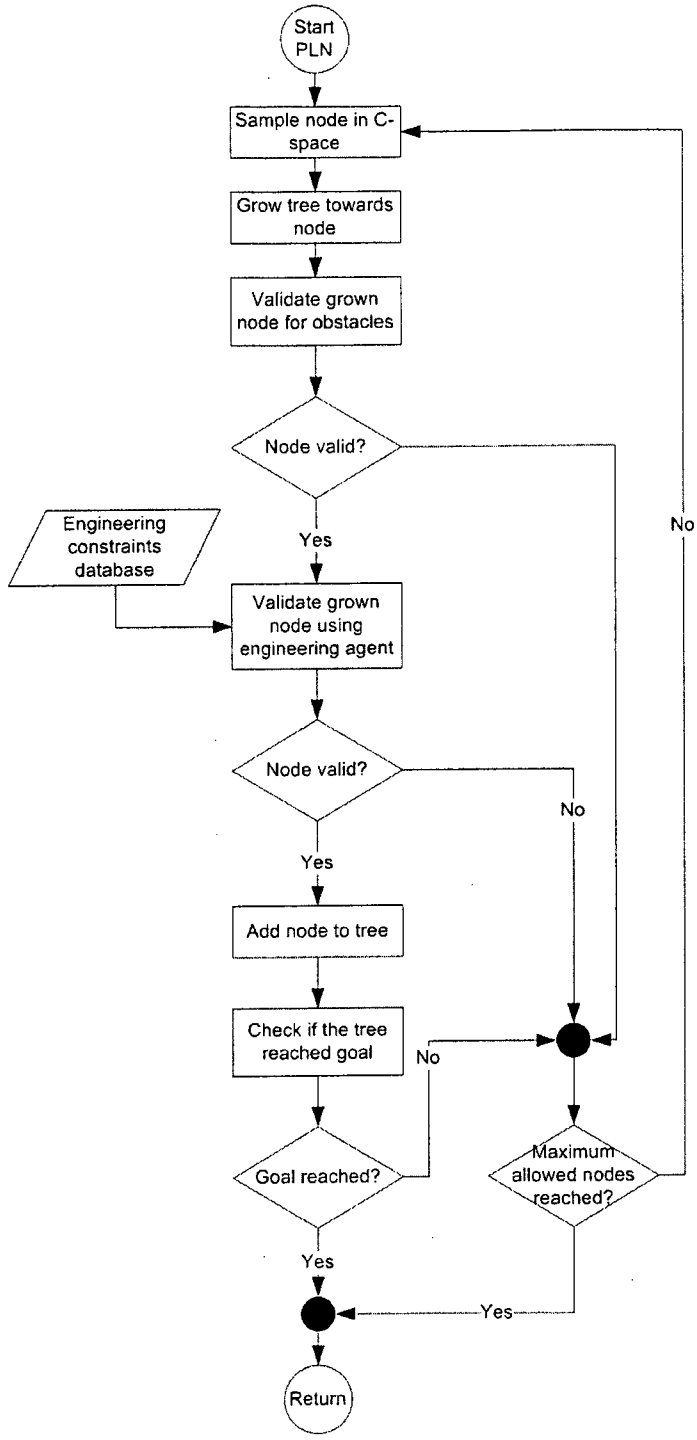
(a) Flow of the general framework

Figure 3.2: Flowchart for the proposed real-time motion planning framework



(b) Motion planning problem definition

Figure 3.2: Flowchart for the proposed real-time motion planning framework (continued)



(c) Planning/re-planning algorithm

Figure 3.2: Flowchart for the proposed real-time motion planning framework (continued)

3.3 KINEMATIC PROPERTIES OF A HYDRAULIC CRANE

To be able to integrate the construction equipment in the framework, a computational model is required that defines two major aspects: the kinematic properties and the geometrical representation of the construction equipment. For the kinematic properties, the construction equipment is considered as a robot, which is composed of a series of links (rigid bodies) connected by joints which allow relative motion of neighboring links. With this robotic model, defining the kinematic properties includes defining: the hierarchal structure of the links, local coordinate systems (frames) and joint types which are either a sliding joint (prismatic) or a rotational joint (revolute). These properties can be defined mathematically in a homogeneous transformation matrix using the DH-notation as explained in Section 2.4. Considering a hydraulic crane as an example of construction equipment, there are two revolute joints (swing of the boom θ_1 and angle to the ground θ_2) and two prismatic joints (boom extension d_3 and cable extension d_4), and on each joint a local coordinate system is attached as presented in Figure 3.3.

Applying DH-notation on a hydraulic crane, we can obtain four transformation matrices, ${}^0T_1$, ${}^1T_2$, ${}^2T_3$,${}^3T_4$ that specify the kinematic relations between the attached axis for each link. ${}^0T_1$ is the homogeneous transformation matrix that transforms the motion from the base coordinate system $\{0\}$ that is attached on the top of the truck to the coordinate system $\{1\}$ which is attached to the bottom of the cabin. Similarly, ${}^1T_2$ transfers the coordinate system from the bottom of the cabin to the base of the boom; ${}^2T_3$ transfers the

coordinate system from the base of the boom to the first boom extension $\{3\}$; finally ${}^{3}T_4$ transfers the coordinate system from the boom extension to the cable $\{4\}$.

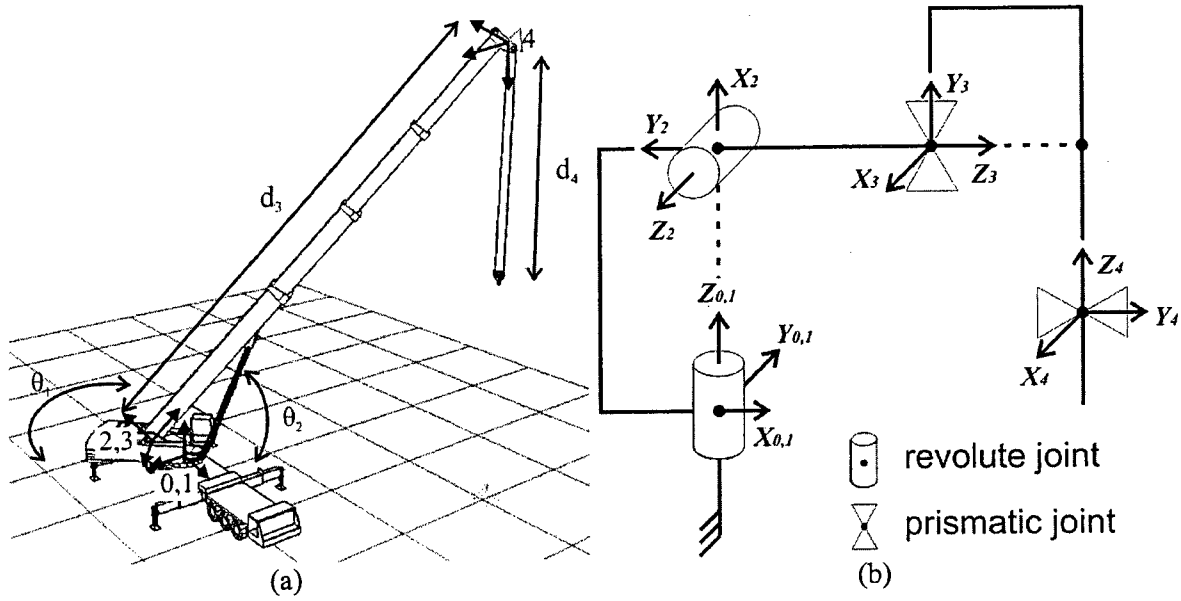


Figure 3.3: Defining the kinematic structure for a hydraulic crane: (a) Frames attached to the crane components; (b) Schematic for the hydraulic crane based on DH-notation

Once the transformation matrix for each link has been developed, the forward kinematics function of the hydraulic crane can be found by multiplying all four link transformation matrices. Since the matrix is essentially transferring the coordinate system from $\{0\}$ to $\{4\}$, we denote it by ${}^{0}T_4$ as shown in Equation 3.1. Appendix C lists the Matlab code used to calculate this matrix.

$$ {}^{0}T_4 = \begin{bmatrix} c\theta_1 c\theta_2 & c\theta_1 s\theta_2 & -s\theta_1 & -d_4 s\theta_1 + d_3 c\theta_1 s\theta_2 \\ s\theta_1 c\theta_2 & s\theta_1 s\theta_2 & c\theta_1 & d_4 c\theta_1 + d_3 s\theta_1 s\theta_2 \\ s\theta_2 & -c\theta_2 & 0 & -d_3 c\theta_2 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad \text{(Equation 3.1)} $$

Where $c\theta_1$ represents $cos(\theta_1)$, $c\theta_2$ represents $cos(\theta_2)$, $s\theta_1$ represents $sin(\theta_1)$, $s\theta_2$ represents $sin(\theta_2)$, θ_1 represents the swing of the boom, θ_2 represents the angle to the ground, d_3 represents the length of the boom, and d_4 represents the length of the cable. The matrix ${}^{0}T_4$ is a homogeneous transformation matrix, which can be used to present both orientation and the position of the cable with respect to the coordinate system $\{0\}$. This matrix representation is the base of the kinematic structure definition for the solver. More discussion about the use of homogeneous transformation matrices to present robot-like machines can be found in (Craig, 2004).

Utilizing DH-notation for modeling the kinematic structure of construction equipments is efficient and optimized since the number of parameters required to define the homogeneous transformation are four instead of six as is the case of affine transformation notation. On the other hand, DH-notation depends on specific assumptions for computing the homogeneous transformation matrix. It depends on having links arranged in a hierarchy where each link is transformed relative to its parent and its frame is attached in a specific way. As a result, any link is transformed based on its parent axis. This conclusion is acceptable for most robotic applications; but for construction equipments there may be cases where it is not acceptable to transform a link relative to its parent. As an example, in a hydraulic crane, the cable cannot be modeled as a direct child to the

boom, because in that case it will be transformed relative to the boom axis instead of being transformed relative to the gravity vector. As the boom rotates, the cable will inherit the boom's rotation instead of having constant rotation aligned to the world axis. Figure 3.4 (a) shows the case where the cable frame {*4*} is parented to the boom frame {*2*} through frame {*3*}. As frame {*2*} is rotating, frames {*3*} and {*4*} are inheriting the rotation.

In this research, a solution is proposed to transform the rotation matrix of the cable relative to the world frame instead to the boom frame while still following the DH-notation. This solution enables the cable to stay oriented towards the gravity vector no matter how the boom rotates. To simplify the process of deriving the solution, the homogeneous transformation matrix of the cable is divided to its main components as shown in the following equation:

$$ {}^{0}T_{4} = \begin{bmatrix} {}^{0}R_{4} & \vdots & {}^{0}P_{4} \\ 0 \;\; 0 \;\; 0 & \vdots & 1 \end{bmatrix} \qquad \text{(Equation 3.2)} $$

Where ${}^{0}R_{4}$ is the 3×3 rotation matrix of {*4*} relative to {*0*}, and ${}^{0}P_{4}$ is the 3-dimensional vector for the position of {*4*} relative to {*0*}. These two components are extracted from the final homogeneous transformation matrix that is computed by multiplying successive homogeneous transformation matrices from frame {*0*} to frame {*4*}. For the cable, to have the rotation matrix transformed relative to the world frame {*W*}, a zero-link (i.e.

$a_{bf}=0$) is introduced with its own frame $\{Z\}$ between the cable and the boom. Thus, the homogeneous transformation matrix will be:

$$ {}^{0}T_{4}={}^{0}T_{1}.{}^{1}T_{2}.{}^{2}T_{3}.{}^{3}T_{Z}.{}^{Z}T_{4} \qquad \text{(Equation 3.3)} $$

This new link is used to provide the cable frame $\{4\}$ with a world-transformed parent frame $\{Z\}$, thus when transforming $\{4\}$ based on the DH-notation, it will be transformed relative to $\{Z\}$ which has a matching orientation to $\{W\}$. Figure 3.4 (b) shows how the new zero-length link frame $\{F\}$ matches the orientation of $\{W\}$ and enables the cable frame $\{4\}$ to be transformed relative to a world oriented frame $\{Z\}$.

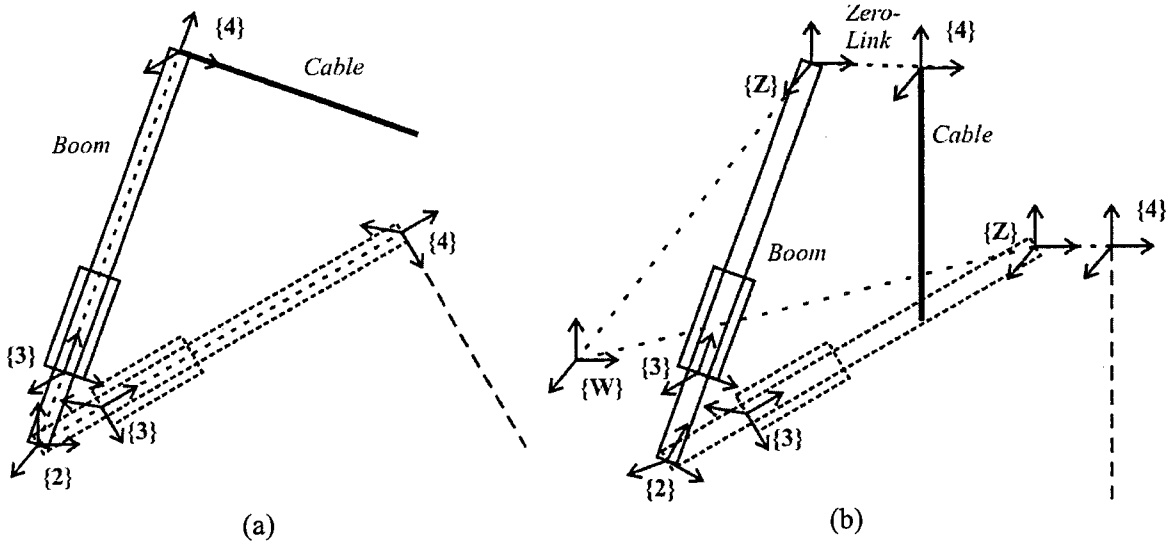


Figure 3.4: The effect of different hierarchy setups on the cable: (a) Cable linked directly to the boom. (b) Cable linked to intermediate frame that compensates for the boom rotation

After adding the new link, the rotation matrix is then composed of:

$$^{0}R_{4}=^{0}R_{1}.^{1}R_{2}.^{2}R_{3}.^{3}R_{Z}.^{Z}R_{4} \qquad \text{(Equation 3.4)}$$

To make the rotation of $\{Z\}$ always match the rotation of $\{W\}$, the rotation of the boom $\{2\}$ should be compensated since the rotation of $\{Z\}$ is affected by $\{2\}$. This compensation is done by computing the inverted rotation matrix:

$$^{3}R_{Z} = [^{1}R_{2}.^{2}R_{3}]^{-1} = {}^{1}R_{3}^{-1} \qquad \text{(Equation 3.5)}$$

Since rotation matrices are orthogonal, the inversion of $^{1}R_{3}$ can be computed by taking its transpose instead computing its inversion as a simplified approach. Thus $^{1}R_{3}^{-1}$ will be

$$^{1}R_{3}^{-1} = [^{1}R_{2}.^{2}R_{3}]^{T} \qquad \text{(Equation 3.6)}$$

Regarding the position vector $^{3}P_{4}$, since the new introduced link is zero-length, then the position vector will remain the same and the homogeneous transformation matrix will be:

$$^{0}T_{4} = {}^{0}T_{3}.\left[\begin{array}{ccc:c} & {}^{1}R_{3}^{T}\,{}^{Z}R_{4} & & {}^{3}P_{4} \\ \hdashline 0 & 0 & 0 & 1 \end{array}\right] \qquad \text{(Equation 3.7)}$$

This solution avoids considering an additional revolute DoF for controlling the cable's orientation; where in that case, the C-space dimension will become a five-dimsion space, and the confguration vector will be $q=(\theta_1, \theta_2, d_3, \theta_4, d_5)$ while the controlable DoFs are only $(\theta_1, \theta_2, d_3, d_4)$. This leads to solve the motion planning problem as if it is a non-honomic, which is not true because the cable motion constraint can be expressed as a configuration constraints as show in Figure 3.5 and Equation 3.8.

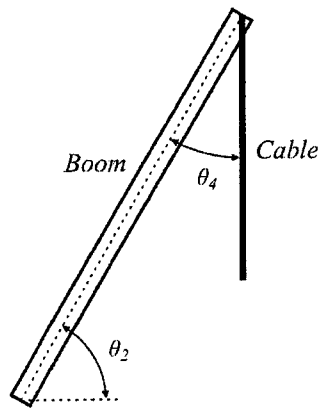


Figure 3.5: Configuration constraint between boom rotation (θ_2) and cable rotation (θ_4)

$$\theta_4 = 90° - \theta_2 \qquad \text{(Equation 3.8)}$$

3.3.1 C-space Dimensionality and Topology

For this research all construction equipment are modeled kinematicly as holonomic robots to avoid using state space instead of C-space. This results in that the C-space dimensionality and topology are affected by the number and types of DoFs of an

equipment. Based on the C-space dimensionality, a configuration in the C-space can be expressed by a vector that has the same number of dimensions of the DoFs of the equipment. For a hydraulic crane that has four DoFs (swing, rotate boom up and down, extend boom and move the hook up and down), its configuration vector should be a 4D vector that lists these DoFs as the following: $q=(\theta_1,\theta_2,d_3,d_4)$. Therefore, the more DoFs considered the more complex the C-space would be. There is strong evidence that the planning solution requires exponential time in the number of dimensions of the C-space, i.e., the number of DoFs of the robot (Kavraki and Latombe, 1998). For a simplified structure of a hydraulic crane, it has four DoFs as introduced previously, but this simplified structure is not used for simulation or control. Instead, a full kinematic definition is used which could have up to seven DoFs, two revolute joints (θ_1 and θ_2) and five prismatic joints (d_3, d_4, d_5, d_6 and d_7), as shown in Figure 3.6.

Solving the problem with full kinematic structure of the crane raises the complexity of the problem. This can be avoided by solving the problem using the simplified structure and then transferring the results to the full kinematic structure for simulation. In the case of the hydraulic crane, the planning results for the boom extension should be transferred from one prismatic joint to four joints for the full structure as shown in Figure 3.7.

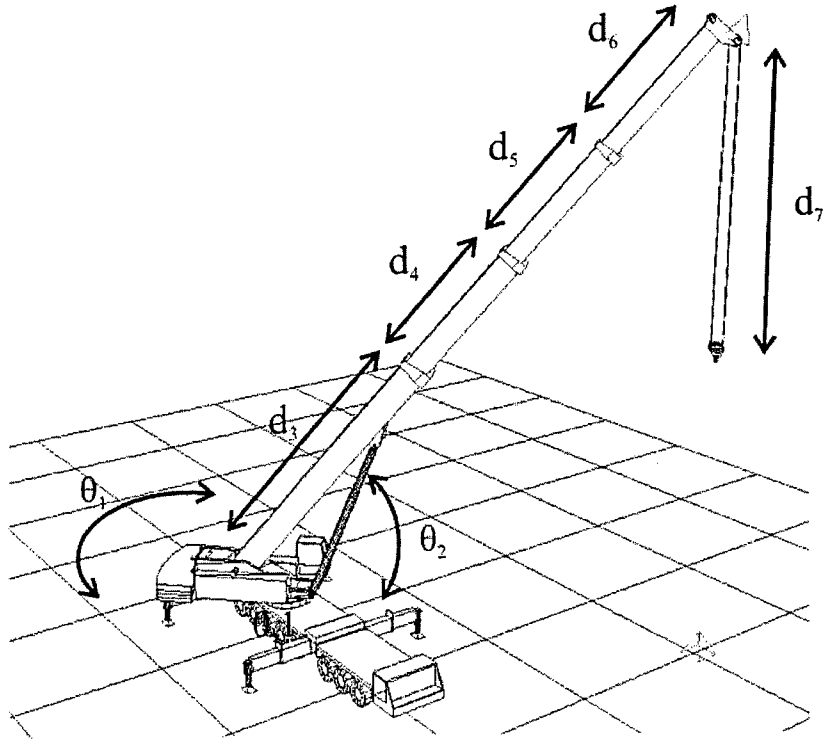


Figure 3.6: Degrees of freedom for a crane

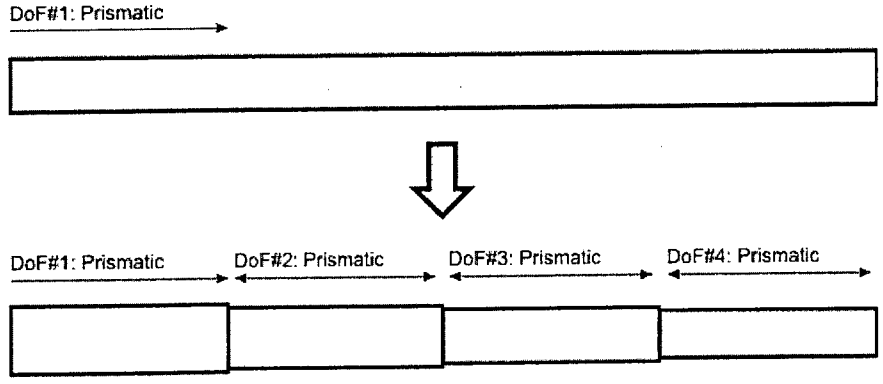


Figure 3.7: Transferring planning results from one prismatic joint to several joints

Transferring the results require deriving a heuristic relationship between the simplified and the full kinematic structure. In the hydraulic crane case, this relationship should

satisfy rules of actions where the boom extends sequentially in the full structure starting from the base part of the boom. The following equation establishes this relationship:

$$q_i = \max\left[\min\left(q_{i-max}, q_{simple} - \left[q_{simple-max} - \sum_{j=1}^{i} q_{j-max}\right]\right), 0\right] \qquad \text{(Equation 3.9)}$$

where:

- q_i: is the extension value of the prismatic joint number i, where i starts from the tip of the boom.
- $q_{i\text{-}max}$: is the maximum value the prismatic joint i can have.
- q_{simple}: is the current value of the joint in the simplified structure.
- $q_{simple\text{-}max}$: is the maximum range of joint limit in the simplified structure. This value should equals to the sum of maximum ranges of all joints in the original structure (i.e. $q_{simple\text{-}max} = \Sigma\, q_{i\text{-}max}$).

As an example for applying the previous equation, Figure 3.8 shows a sample case where it is required to transfer the current extension value from the simplified boom to the full one based on Equation 3.8.

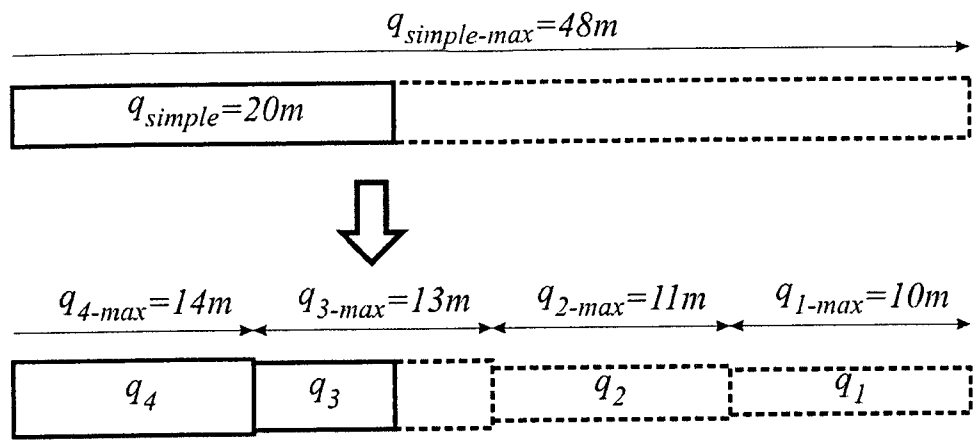


Figure 3.8: Example of transferring planning results from one prismatic joint to several joints

The values for all joints are then computed as in the following:

$$q_1=max[min(10,20-[48-(10)]),0]=0m \quad \text{(Equation 3.10)}$$

$$q_2= max[min(11,20-[48-(10+11)]),0]=0m \quad \text{(Equation 3.11)}$$

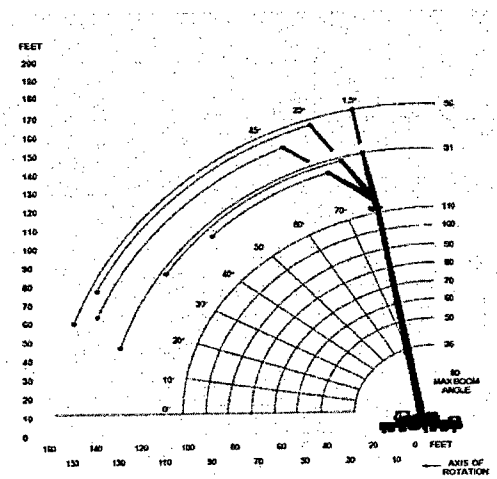
$$q_3= max[min(13,20-[48-(10+11+13)]),0]=6m \quad \text{(Equation 3.12)}$$

$$q_4= max[min(14,20-[48-(10+11+13+14)]),0]=14m \quad \text{(Equation 3.13)}$$

For the topology of construction equipment, implicit representation of the topology of the C-space is avoided in this research because construction equipment usually is composed of a kinematic chain of different types of joints. One way to avoid implicit C-space representation is to depend on randomized sampling to capture the C-space without the need to derive its topology. This decision directs the selection of the planning and re-planning algorithms towards the sampling-based algorithms that will be discussed in Section 3.6.

3.4 ENGINEERING CONSTRAINTS

To ensure realistic planning and re-planning for construction equipment, engineering constraints should be considered in the framework to define the planning problem. Taking cranes as an example, the engineering constraints of cranes are mainly from working ranges and load charts. Working ranges show the minimum and maximum boom angle according to the length of the boom and the counterweight (Figure 3.9). Load charts give the lifting capacity based on the boom length, boom angle to the ground and the counterweight. These data of engineering constraints are all stored in databases that can be accessed later when generating paths.



(a)

(Feet)	36	50	*60	70
10	+140,000 (68)	109,500 (75)	84,200 (78)	**56,450 (80)
12	110,500 (64)	104,500 (72.5)	79,850 (76)	56,450 (78.5)
15	96,800 (58.5)	91,400 (69)	73,900 (73)	56,450 (76)
20	78,750 (47)	75,300 (62)	59,600 (67.5)	56,450 (71.5)
25	59,800 (32.5)	59,750 (55)	50,000 (62.5)	48,900 (67)
30		47,300 (47)	42,300 (56.5)	41,900 (62.5)
35		38,550 (37.5)	36,950 (50)	36,400 (57.5)
40		28,450 (24.5)	28,450 (43)	29,700 (52)
45			23,400 (34.5)	24,650 (46.5)
50			19,450 (23)	20,700 (39.5)

(b)

Figure 3.9: Working range (a) and load chart (b) of a crane (Groove Crane, 2006)

Al-Hussain et al. (2000) described the development of a comprehensive database (called D-CRANE) designed to support efficient selection of cranes that includes operational information about crane geometry, lift configurations, lift capacity settings, accessories,

and attachments. In this research, the load chart data of the case study crane is added to a simplified database to test the developed prototype system as will be explained in Section 4.4.2.

In the proposed framework, engineering constraints should be considered through all motion planning steps in order to guarantee safe paths in terms of stability of the equipment along the entire length of the path. This requirement leads to a planning process that is interwoven with engineering constraints validation. While the solver is taking decisions for the next step of the path it is generating, engineering constraints are validated and included as additional decision factors for the solver. Validating engineering constraints is based on a search algorithm that is developed to validate solver queries by searching for the category that satisfies the query in the specified data cluster. If the algorithm finds this category, a decision of acceptance is returned to the solver for the stated query, otherwise it will reject the query and the solver has to try a different strategy that satisfies the engineering constraints. This algorithm is implemented based on a brute-force search. It simply loops on all values in the data cluster and does multi-comparison queries between the current and the queried values. At the end, it returns either *true* or *false* based on the searching results. This algorithm can be enhanced in terms of search efficiency, but since the size of the data set is relatively small, the effect of this enhancement is not noticeable and the current algorithm is sufficient for such searching problems. Figure 3.2 (c) shows the flowchart for the planning phase considering engineering constraints.

In this research, this searching algorithm is referred to as Local Validation Algorithm (LVA) that it is applied using the database that contains the engineering constraints. Figure 3.10 shows pseudo code for the simplified searching algorithm. This algorithm works by accessing first the data set from the database that matches the specified Outriggers and Counterweight parameters (line 1), it then prepares searching variables by converting simulation scene inputs to compatible units with of the data stored in the database (lines 2-4). The first phase of searching starts by enumerating all available extension groups to specify the group that encompasses the current state of boom extension (lines 5-11). The second phase of search is done for specifying the load group under the specified extension group after inverting the set to be able to search in reverse order (lines 12-17). In case both searching phases succeeded, LVA will return *true* to the solver for the current boom configuration, otherwise it will return *false* and the current state will not be included in crane's path.

LVA(*Outriggers, Counterweight, BoomExtension_State, BoomAngleToGround_State, Load, Capacity_Reduction*)

```
1.  data = queryDataSet(Outriggers, Counterweight);
2.  d = convertToFeets(BoomExtension_State);
3.  th = convertToDegrees(BoomAngleToGround_State);
4.  load_mod = Load * (100 + Capacity_Reduction)/100;
5.  repeat for each data:extension_class(i) in data:extension classes
6.    if (d≤ data:extension class(i))
7.      extension_class_id = i+1;
8.    else
9.      break;
10. if (extension_class_id = null)
11.   return false;
12. invertSort(data:extension_class_id(i));
13. repeat for each data:load_class(j) in data:extension_class_id(i)
14.   if (load_mod ≤ data:load_class(j))
15.     load_class_id = j;
16.     break;
17. if (load_class_id = null)
18.     return false;
19. return true;
```

Figure 3.10: LVA pseudo code

3.5 GEOMETRICAL REPRESENTATION

Accurate geometrical representation for the construction equipment is required not only to render and visualize the results in a 4D simulated environment, but it is essential for performing collision detection calculations while creating the motion paths. As discussed in Section 2.2, polygonal meshes are preferred for such tasks, where real-time rendering and collision detection is done more efficiently. Processing polygonal meshes starts by tessellating them to a simple mesh of triangles. Figure 3.11 shows a 3D model for a hydraulic crane tessellated into triangles for 3D real-time rendering.

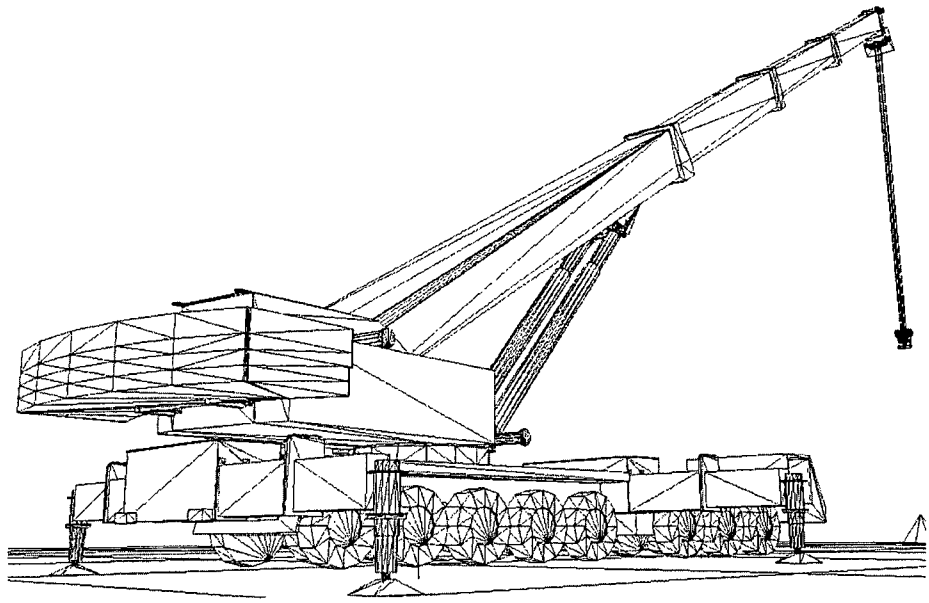


Figure 3.11: 3D tessellated model for a hydraulic crane

Additional processing is done that encompasses merging and linking polygonal data, as well as deriving new data, such as normals, for surface shading and collision detection. On top of these operations in the processing phase, another common operation called orientation is needed to make sure that all polygons forming a surface are made to face the same direction. Once orientation is performed, checking whether a mesh forms a solid surface is important for its use by a number of different algorithms including collision detection that is executed during the path planning. For the case studies introduced in Section 4.6, the previous process is applied on 3D model of hydraulic crane and tower crane to have compatible models with the proposed APIs and libraries for real-time rendering and collision detection calculations.

3.5.1 Increasing Safety by Avoiding Semi-Free Paths

When doing motion planning for construction equipment, safety is one of the major concerns that need to be considered and enhanced in the framework. Increasing the safety of the generated paths can be accomplished by avoiding semi-free paths. A path is considered semi-free if the equipment touches obstacles without overlapping. Topologically, assuming that C is the set of the C-space and C_{free} is the open set of the free space, semi-free paths are contained in a semi-free space which is a closed subset of C and its boundary is a superset of the boundary of C_{free} (LaValle, 2006).

One way to ensure free paths can be done by adding a safety buffer uniformly around obstacles. Dilated obstacle can be expressed generally as a type of convolution using Minkowski sum (Choset et al., 2005). Let A and B be two point sets, and a and b be the position vectors corresponding to pairs of points in A and B. The Minkowski sum is then defined as:

$$A \oplus B = \{a + b \in R^n \mid a \in A \text{ and } b \in B\} \qquad \text{(Equation 3.14)}$$

Where $a+b$ is the vector sum of the position vectors a and b. Visually, the Minkowski sum can be seen as the region wept by A translated to every point in B (or vice versa) (Ericson, 2005).

Figure 3.12 shows the effect of dilation with different values around a polygonal obstacle and the closed boundary: (a) Polygonal obstacle and boundary without any dilation, the two configurations can be connected with feasible paths either above or below the obstacle, (b) dilation causes the obstacle to expand and boundary to shrink which eliminates all possible paths below the obstacle, (c) bigger dilation radius disconnect the connectivity between the two configurations and no feasible paths can be returned.

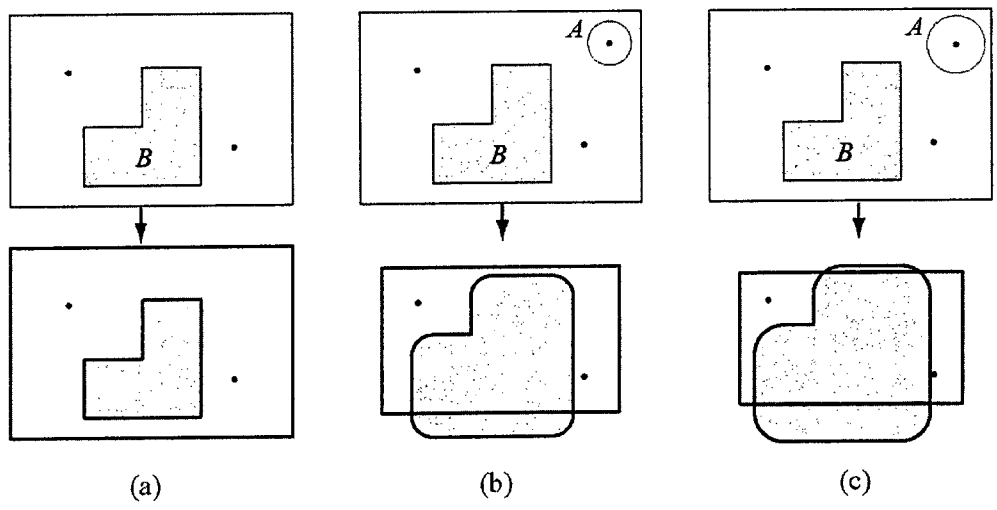


Figure 3.12: Applying different dilation values to a simple 2D obstacle (Adopted from Choset et al., 2005)

As noted in the previous figure, dilation in constrained spaces can sometimes waste good paths or can return no paths at all even if there was one. This is because when doing dilation around "difficult" (narrow) parts in the space, some paths will be eliminated. Even though, this behavior is acceptable in construction equipment applications, because

it is preferable to have no feasible path than unsafe paths that go between tight obstacles. There is no specific value for the clearance, as an example, in the research of Al-Hussein (2005) a clearance value around the crane boom is specified to be around 0.2 meters.

3.5.2 Critical Volumes

Safety is one of the major concerns of this research, to ensure safe equipment planning, certain volumes in the construction space are considered unsafe if certain construction equipment movement paths intersect them. Figure 3.13 shows an example of such volumes where the path of one crane goes through the space between the boom and the cable of another crane. Such cases are not desirable for safety issues and it is better to avoid generating paths in such volumes (Fair, 1998; British Standard, 2000). In this research, volumes that are applicable but not safe are called critical volumes.

The proposed approach to avoid generating paths that intersect with these volumes is to construct 3D surfaces from 3D points that define the critical volume's extents. These constructed surfaces can be used later as obstacles to prevent the solver from generating paths that go through those critical volumes. These 3D surfaces should be recomputed based on the spatial state that is updated based on other dynamic elements in the scene. This leads to reconstructing these volumes dynamically as the spatial state is updating. Thus, it is necessary to consider generated surfaces as dynamic obstacles that are detected in real-time. Figure 3.14 shows specified points for a hydraulic crane.

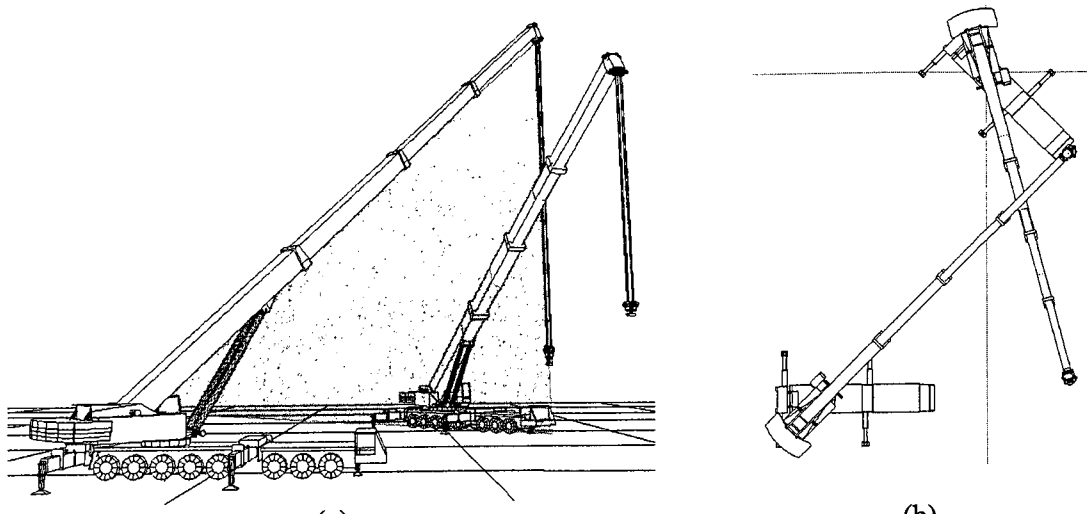


Figure 3.13: The path of one crane goes through the critical volume that is bounded under the boom and the cable of the other crane: (a) perspective view; (b) top view

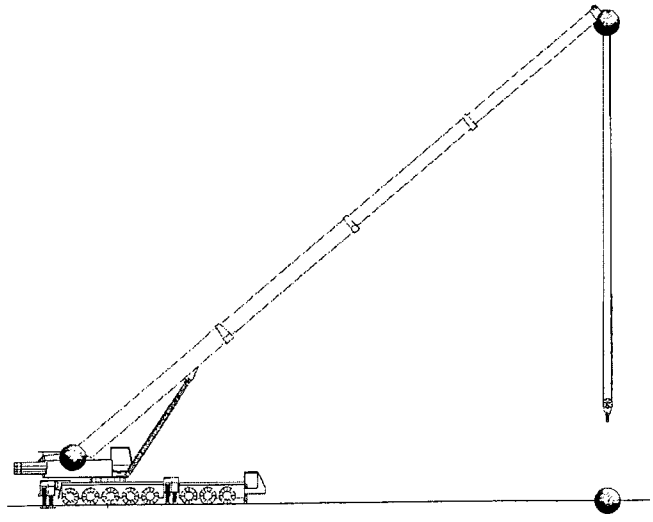


Figure 3.14: Points attached on the crane for defining a critical volume

As explained in Section 2.2, several algorithms are available for constructing 3D surfaces from point clouds. In this research points that define the critical volume are located only on the extents of the volume and they are added in specific order, thus it is possible to generate the 3D surface by creating a polygonal surface that connect these points in their specified order and avoid implementing general 3D surface constructing algorithms.

3.6 THE SOLVER

After generating the feasible C-space, path planning becomes a problem of finding a path that connects the initial configuration to a particular goal configuration, which is the responsibility for the core component of the framework: the solver. For motion planning of construction equipment, the planning algorithm should consider the following requirements:

1. *Efficiency*: Path planning has been proven a hard problem (Reif, 1979). In the last decade, more interest has grown in developing practical path planners (Latombe, 1991), (Barraquand, 1997). These planners embed weaker notions of completeness (e.g., probabilistic completeness) and/or can be partially adapted to specific problem domains in order to boost performance in those domains. In re-planning, efficiency is the most important factor, because decisions usually need to be taken in real-time to cope with the dynamic nature of the environment.

2. *Safety*: enhancing the safety of the generated paths can be increased with a planning algorithm that tends to maximize clearance around obstacles. The dilation step in the C-space generation (Section 3.5.1) ensures safety of the paths but it can waste existing paths. However, having a planning algorithm that avoids semi-free paths is preferable.
3. *Applicability*: to ensure generating realistic paths, the algorithm needs to consider construction equipment rules of action. Rules of actions are based on expert rules, such as avoiding combinations of hoisting and swinging or hoisting and luffing at the same time; and preventing the boom's motion when a crane is traveling. Thus, rules of actions restrict equipment movement to one DoF for each step. Generating paths that satisfy this constraint requires the planning algorithm to generate paths where all of its components are decomposed along only one dimension in the C-space.
4. *Optimality*: not all algorithms guarantee to generate optimal paths in terms of length, time or cost. Thus, it is important to consider this factor in the investigated algorithm.

The planning algorithm that is used in this research is based on the RRT algorithm. Enhancement and modifications are made to make it suitable for construction equipment applications considering the above requirements. The following subsection describes the naïve algorithm and its modifications.

3.6.1 Planning Algorithm

The essential RRT algorithm is outlined in Figure 3.15 (LaValle, 1998). Beginning with the initial configuration as the root node, it incrementally grows a tree until the tree reaches the goal configuration. To grow the tree, (lines 2-8 in function *GrowRRT*), first a target configuration q is randomly selected from the configuration space using the function *ChooseTarget*. Then, a *NearestNeighbor* function selects the node q_{near} in the tree closest to q. Finally, a new node q_{new} is created in an *Extend* function by growing the tree some distance ε from q_{near} towards q as shown in Figure 3.16. If extending the tree towards q requires growing through an obstacle or through a state rejected by the engineering agent, no extension occurs. This process is repeated until the tree grows to within some user-defined threshold of the goal (line 3). A very nice property that follows from this method is that the tree growth is strongly biased towards unexplored areas of the configuration space. Consequently, exploration occurs very quickly.

Several choices for the steps of the above algorithm are still unspecified and need to be modified to fit the construction equipment application. In particular, we need to define how random configurations are created in line 4, clarify the notion of a candidate neighbor in line 5 and the *Extend* function in line 6.

InitRRT()
1. T.add(q_{init});

GrowRRT()
2. $q_{new} = q_{init}$;
3. while (**Distance**(q_{new}, q_{goal}) > distance-threshold)
4. q = **ChooseTarget()**;
5. q_{near} = **NearestNeighbor**(q);
6. q_{new} = **Extend**(q_{near}, q);
7. if($q_{new} \neq$ null)
8. T.add(q_{new});

ChooseTarget()
9. return **RandomNode**();

Main()
10. **InitRRT**();
11. **GrowRRT**();

Figure 3.15: The basic RRT algorithm (Adapted from LaValle, 1998)

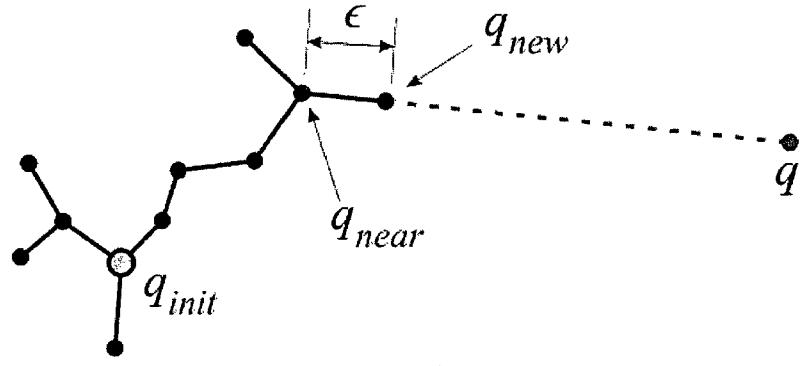


Figure 3.16: The *Extend* function (Kuffner and LaValle, 2000)

Random Configuration

The nodes of q should constitute a rather uniform random sampling of C_{free}. Every such configuration is obtained by drawing each of its coordinates from the interval of allowed values of the corresponding DoF using the uniform probability distribution over this interval. For the hydraulic crane as an example, four random values are drawn between the ranges of each DoF to generate a random sample q.

Nearest Neighbor and Metric in C-space

Virtually all sampling-based planning algorithms require a function that measures the distance between two points in C (LaValle, 2006). In the RRT algorithm that is presented in Figure 3.15, a metric function is required when calculating the distance between two configurations in C (line 3), and when searching for the nearest neighbor node in C (line 5). It is straightforward to define Euclidean distance function in R^n which works well for C-spaces that is generated from prismatic joints only. However, construction equipment has different combinations of prismatic and revolute joints (Section 3.3 explains the joints types for a hydraulic crane). Thus, to define a distance function over any C is more challenging since that the topology of C can be any combination of R^n and T^n based on the joint type as explained in Section 2.5.2. For a simple case of one revolute joint, C is represented by a circle (S^1) or in some references called matrix group *SO(2)* (Choset et al. 2005). In this C-space, when calculating the distance between angles it is noticed that the Euclidean metric for R^n does not give the distance traveling along the circle. It instead takes a shortcut by computing the length of the line segment in R^n that connects the two

points, which is inaccurate. An alternative metric is obtained by directly required to calculate the distance between angles θ_1 and θ_2. However, in this case special care has to be given to the identification, because there are two ways to reach θ_2 from θ_1 by traveling along the circle. This causes a min to appear in the metric definition: $\rho(\theta_1, \theta_2) = min\ [|\theta_1 - \theta_2|, 2\pi - |\theta_1 - \theta_2|]$, for which $\theta_1, \theta_2 \in [0, 2\pi]/\sim$.

For more general cases, like metrics for matrix group *SE(2)* where the C-space is $R^2 \times SO(2)$ (Choset et al., 2005), it can be formed by applying the Cartesian product rules to a metric for R^2 and a metric for *SO(2)*, Cartesian products of metric spaces extend nicely across Cartesian products, which is very convenient because C-spaces are often constructed from Cartesian products, especially in the case of multiple links. Let (X, ρ_x) and (Y, ρ_y) be two metric spaces for any two links in the construction equipment. A metric space (Z, ρ_z) can be constructed for the Cartesian product $Z = X \times Y$ by defining the metric ρ_z as $\rho_z(z, z') = \rho_z(x, y, x', y') = c_1\rho_x(x, x') + c_2\rho_y(y, y')$, in which c_1 and c_2 are any positive real constants, and $x, x' \in X$ and $y, y' \in Y$. Each $z \in Z$ is represented as $z = (x, y)$. Other combinations lead to a metric for Z; for example, $\rho_z(z, z') = (c_1[\rho_x(x, x')]^p + c_2[\rho_y(y, y')]^p)^{1/p}$ is a metric for any positive integer p. Once again, two positive constants must be chosen. It is important to understand that many choices are possible, and there may not necessarily be a "correct" one. For example in the case of construction equipment with a revolute and a prismatic joints, if $c_1 = c_2 = 1$, the range for S^1 is $[0, 2\pi)$ using radians but $[0, 360)$ using degrees. If the same constant c_2 is used in either case, two very different metrics are obtained. The units applied to R^2 and S^1 are completely incompatible.

As noticed previously, for construction equipments it is necessary to define a metric not only as Euclidean metric, which is the familiar Euclidean distance in R^n, but also a metric for comparing angles in case of rotational joints. Also, it is important to be able to combine different metrics for solving general cases where the equipment kinematic definition includes both types of joints: prismatic and revolute with different ranges. In that case, the problem of relating different kinds of quantities arises. For example, in the case of a hydraulic crane, the C-space is composed of $R^2 \times T^2$ and the metric defined must compare two distance and two angular quantities with different ranges. This requires specifying four different constants (c_1, c_2, c_3, c_4) to create a balanced metric along all C-space dimensions. Thus the metric will be:

$$\sqrt{\sum_{i=1}^{n} c_i m_i^2} \qquad \text{(Equation 3.15)}$$

where:

- n is the number of links, for a hydraulic crane $n=4$.
- c is a positive constant for each DoF (q).
- m for prismatic joints is $m=|d-d'|$.
- m for revolute joints is $m= min\ [|\theta - \theta'|, (2\pi - |\theta - \theta'|)]$.

Specifying the weights coefficients (c_i) that balance the metric along all dimensions in the C-space is a complex optimization problem that depends on the type of all DoFs and their ranges. Thus, these weights should be recomputed for every different kinematic structure.

In this research, a novel approach is presented to calculate these weights for any combination of DoFs and for any variety of ranges. The concept for computing these coefficients depends on calculating a speed value ($\partial q/\partial t$) along each dimension that enables the RRT to reach all boundaries of each dimension at the same time. For calculating this speed, an abstract hypercube with the same C-space dimensionality is introduced. This hypercube is subdivided to a constant number of voxels that have a unified size along all dimensions in C-space. The speed for each DoF can be then calculated by dividing the range of that DoF by the number of voxels along that dimension. This concept is applied in the following equations to calculate the speed for revolute and prismatic DoFs.

1. For prismatic joints:

$$\partial d/\partial t = |d_{max} - d_{min}|/hcube_{subs} \qquad \text{(Equation 3.16)}$$

2. For revolute joints:

$$\partial r/\partial t = min\,[|\theta_{max} - \theta_{min}|,\ (2\pi - |\theta_{max} - \theta_{min}|)]/hcube_{subs} \qquad \text{(Equation 3.17)}$$

where:

- $\partial d/\partial t$ and $\partial r/\partial t$ are the speed values for a prismatic joint and a revolute joint, respectively.
- d_{max} and d_{min} are the maximum and minimum range for a prismatic joint, respectively.
- $\theta_{max} - \theta_{min}$ are the maximum and minimum range for a revolute joint.
- $hcube_{subs}$ is the subdivision number of the hypercube along any dimension.

Enabling the RRT to reach all boundaries of each dimension at the same time guarantees a balanced RRT over all dimensions that have a uniform coverage of the C-space. This is a critical issue when RRTs are searching the C-space for paths, because unbalanced RRTs will be unable to find paths in unexplored space.

Figure 3.17 shows a visual representation of the RRT generated in the C-space of a hydraulic crane using the proposed metric discussed previously (a) and a metric with predefined weights (b). In this case, only the first three DoF of the C-space can be visualized in a 3D view. These DoFs are the swing (θ_1), boom angle to the ground (θ_2) and boom extend (d_3). Each of these DoFs has a different range where the range for θ_1 = (-180°,180°), θ_2 = (0°,90°) and for d_3 = (10.8m,33m). Using the proposed metric, RRT was grown in all dimensions in a balanced way that permits the tree to cover the entire C-space. The balancing along all dimensions was generated by computing the metric weights automatically based on a hypercube with 150^3 voxels.

In case of using predefined metric weights (c_1=1, c_2=0.25, c_3=0.5 corresponding to θ_1, θ_2 and d_3, respectively), RRT was grown poorly along the θ_2 dimension, which prevents the tree from sampling the C-space efficiently.

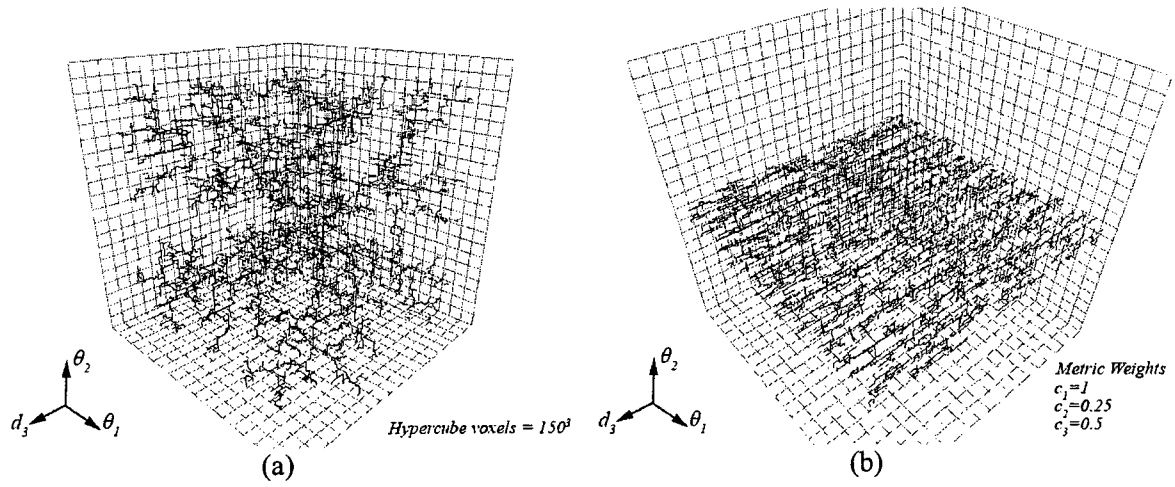


Figure 3.17: RRT visualization for the first 3DoF of a hydraulic crane: (a) Balanced RRT using the hypercube to calculate metric weights; (b) Unbalanced RRT using predefined weights for the metric

The *Extend* function

The *Extend* function is responsible for moving towards the randomly generated point with a small distance ε. The issue of the *Extend* function with engineering rules of actions is that the extension is made generally over multiple dimensions in the C-space, which means that the equipment will move more than one DoF at the same time. This is not allowed for construction equipments since it conflicts with its rules of actions. Thus, the

Extend function should be modified to accommodate this requirement as will be described in Section 3.6.2.

3.6.2 Enhancing the Algorithm

Several extensions to the basic algorithm tend to improve the speed of the search and make it suitable for construction equipment planning. Firstly, the basic algorithm uniformly grows the tree by always selecting q randomly. However, efficiency can be increased by biasing the search towards the goal: in the *ChooseTarget* function, let q be the goal with probability p and choose it randomly with probability $1-p$. As p increases, the RRT behaves increasingly like *bestfirst* search (Ferguson et al, 2006). Figure 3.18 shows the modified version of the *ChooseTarget()* function:

```
ChooseTarget()  1. p = RandomReal([0.0, 1.0]);
  2. if (p < biasing_probability)
  3.    return q_goal;
  4. else
  5.    return RandomNode();
```

Figure 3.18: Including the goal-biasing into the *ChooseTarget()* function

Additionally, better convergence to a solution can be achieved by implementing the *Connect* heuristic (Kuffner and LaValle, 2000) that attempts to move over a longer distance. The *Connect* heuristic is a greedy function that can be considered as an alternative to the *Extend* function in Figure 3.15. Instead of attempting to extend an RRT by a single ε step, the *Connect* heuristic iterates the *Extend* step until q or local/global constraint is reached, as shown in the *Connect* function description in Figure 3.19. This operation serves a similar function as the artificial potential function in a randomized potential field approach. In both cases, the heuristic allows rapid convergence to a solution.

Connect (q_{near}, q)
1. repeat until ($q_{new} = q$ or q_{new} = in collision or **engCns**(q_{new}) = false)
2. q_{new} = **Extend** (q_{near}, q);
3. return q_{new};

Figure 3.19: The *Connect* function

The previous *Connect* function naturally extends towards the sampled node along multiple dimensions simultaneously. As mentioned previously, this is not allowed in the case of construction equipments, as this behavior conflicts with the rules of action. To resolve this issue, a modified version of the *Connect* function is used where instead of extending along multiple dimensions to reach the sampled node itself, the extension will only happen along the best dimension that leads the extension to reach the nearest position to the sampled node. Thus, the *Connect* function will stop extending when the

distance between the extension (q_{new}) and the sampled node (q) is getting bigger. Figure 3.20 shows the concept of modified function that adapts construction equipments rules of action in case of 2 DoFs and Figure 3.21 shows the pseudo code for it.

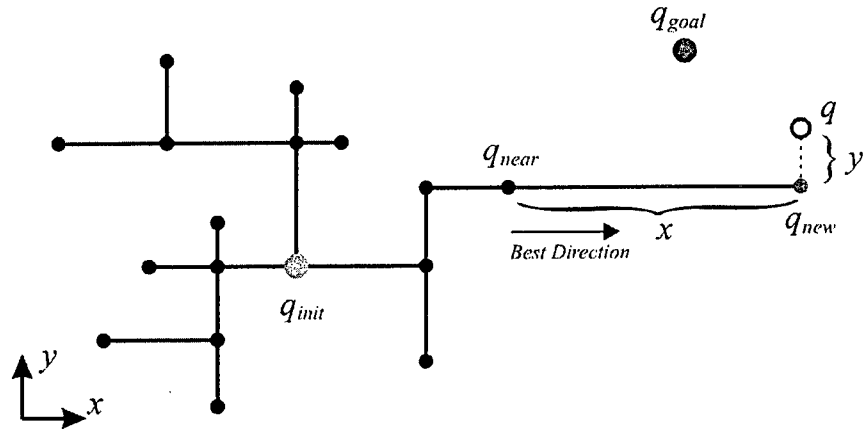


Figure 3.20: The concept of the rules of action adapted *Connect* function

Connect (q_{near}, q)
1. $qDist = 0$;
2. repeat until (**Distance**(q_{new}, q) > $qDist$ or q_{new} = in collision or **engCns**(q_{new}) = false)
3. $qDist$ = **Distance**(q_{new}, q);
4. q_{new} = **Extend** (q_{near}, q);
5. return q_{new};

Figure 3.21: Adding rules of action modification to the *Connect* function

From an engineering point of view the *Connect* function is better than the *Extend* function because the generated path with the *Connect* function is composed from the minimum amount of nodes, which means the minimum amount of sub-tasks for the equipment to accomplish its task. On the other hand, paths generated with the *Connect*

function tend to be longer and have unnecessary steps than those generated using the *Extend* function. This is because of the greedy nature of the *Connect* function that tries to reach the sampled node as much as it can. To get the best of the *Connect* and the *Extend* functions, a modified version of the *Connect* function can be implemented to enhance the optimality of the algorithm and reduce redundancy in the path. To achieve this, an additional check is introduced in the *Connect* function to limit its extension when the projection of the extended node on the perpendicular dimension reaches the goal. Figure 3.22 lists the modified *Connect* function that is called *Limited Greedy Connect*.

Connect (q_{near}, q)
1. *qDist = 0;*
2. repeat until (**Distance**(q_{new}, q) > *qDist* or
   q_{new} = in collision or
   (**isExtendingTowardsGoal** = true and **Distance**(q_{new}, q_{goal}) > *gDist)* or
   engCns(q_{new}) = false)
3. *gDist* = **Distance**(q_{new}, q_{goal});
4. *qDist* = **Distance**(q_{new}, q);
5. q_{new} = **Extend**(q_{near}, q);
6. return q_{new};

isExtendingTowardsGoal(q_{near}, q, q_{goal})

7. q_{new} = **Extend**(q_{near}, q);
8. if (**Distance**(q_{new}, q_{goal}) < **Distance**(q_{near}, q_{goal}))
9. return true;
10. else
11. return false;

Figure 3.22: Pseudo code for the *Limited Greedy Connect*

In other words, the *Connect* function continues to extend towards the sampled node as long the distance between the extended node and the goal is shrinking. Figure 3.23 shows the behavior of the modified *Connect* function.

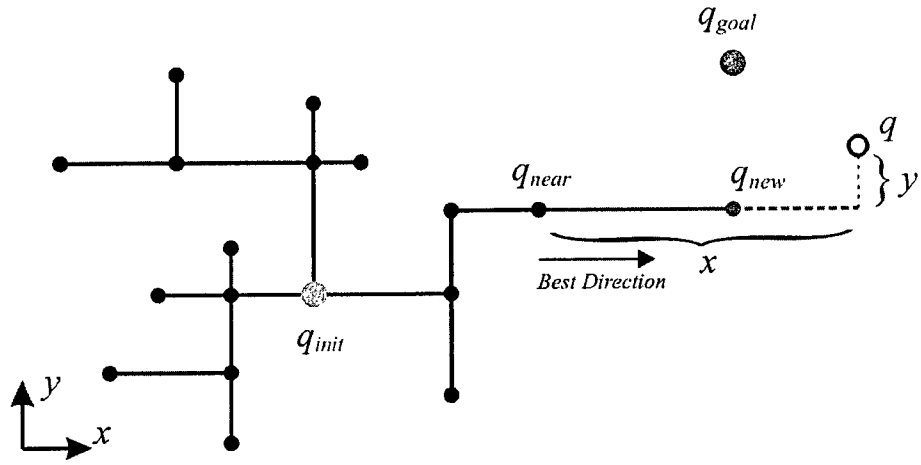


Figure 3.23: The behavior of the *Limited Greedy Connect* function

Combining all the previous modifications, the final algorithm will be as listed in Figure 3.24. The algorithm name for the combination of the new proposed modifications is called *RRTBiasedLimCon* which indicates a goal biased limited greedy connect RRT.

ChooseTarget()
1. p = **RandomReal**([0.0, 1.0]);
2. if (p < biasing_probability)
3. return q_{goal};
4. else
5. return **RandomNode**();

Connect (q_{near}, q)
6. $qDist = 0$;
7. repeat until (**Distance**(q_{new}, q) > $qDist$ or
   q_{new} = in collision or
   (**isExtendingTowardsGoal** = true and **Distance**(q_{new}, q_{goal}) > $gDist$) or
   engCns(q_{new}) = false)
8. $gDist$ = **Distance**(q_{new}, q_{goal});
9. $qDist$ = **Distance**(q_{new}, q);
10. q_{new} = **Extend**(q_{near}, q);
11. return q_{new};

isExtendingTowardsGoal(q_{near}, q, q_{goal})

12. q_{new} = **Extend**(q_{near}, q);
13. if (**Distance**(q_{new}, q_{goal}) < **Distance**(q_{near}, q_{goal}))
14. return true;
15. else
16. return false;

InitRRT()
17. T.add(q_{init});

GrowRRTBiasedLimCon()
18. q_{new} = q_{init};
19. while (**Distance**(q_{new}, q_{goal}) > distance-threshold)
20. q = **ChooseTarget**();
21. q_{near} = **NearestNeighbor**(q);
22. q_{new} = **Connect**(q_{near}, q);
23. if($q_{new} \neq$ null)
24. T.add(q_{new});

Main()
25. **InitRRT**();
26. **GrowRRTBiasedLimCon**();

Figure 3.24: *RRTBiasedLimCon* algorithm

3.6.3 Multi-Equipment Planning/Re-Planning Algorithm

The *RRTBiasedLimCon* algorithm discussed in the previous subsection is able to efficiently provide solutions to problems involving vast, high-dimensional configuration spaces. However, this algorithm is unable to cope with new information concerning the environment. As discussed in Section 1.1, initial information regarding the environment is rarely perfect. Therefore, it is important that the algorithm is able to update solutions to the dynamically changing environment over time. In such scenarios, RRT-based approaches typically abandon the current solution and grow a new RRT from scratch. This can be a very time-consuming operation, particularly if the planning problem is complex. In this subsection, a re-planning algorithm is investigated for repairing RRTs when new information concerning the configuration space is received. Instead of abandoning the current RRT entirely, the proposed approach efficiently removes just the newly-invalid parts and maintains the rest (trimming the tree). It then grows the remaining tree until a new solution is found (re-growing the tree). The concept of this algorithm was first presented in (Ferguson, 2006) as Dynamic Rapidly-exploring Random Trees (DRRT). It has been modified in this research to fit the proposed *RRTBiasedLimCon* algorithm for the dynamic framework.

Pseudo code for the modified DRRT is presented in Figure 3.25. The algorithm starts with the *Main()* function by inverting the growing direction of the initial tree to be from the goal to the initial configuration (line 16). This modification allows maintaining more of the original tree during repairing, as only the tips of the tree will be affected by newly-

observed obstacles. Otherwise, the root of the original tree would constantly be changing and so the entire tree would need to be re-grown. Further, since observations are typically being made in the vicinity of the equipment (through onboard sensors), this modification allows to maintain more of the previous tree during repair, as only the tips of the tree will be affected by newly-observed obstacles. As the robot starts executing the path generated by the *GrowRRTBiasedLimCon()* function (line 18), when an obstacle is detected in the C-space, the *InvalidateNodes()* function is called. In this function, first the edges in the current tree that intersect this obstacle are found (line 12). Each of these edges will have two endpoint nodes in the tree: one will be the parent of the other in the tree. In other words, one of these nodes (the parent) will have added the other (the child) to the tree through the proposed *Connect* function. The child endpoint node of each edge is then marked as invalid (line 15). After all the child endpoint nodes of the affected edges have been marked, the solution path is checked for invalid nodes (line 25). If any are found, the tree needs to be re-grown through the *RegrowRRTBiasedLimCon()* function which is based on the *GrowRRTBiasedLimCon()* function explained in Figure 3.24. This involves trimming the tree (*TrimRRTBiasedLimCon()* function) and growing the trimmed tree out to the goal (*GrowRRTBiasedLimCon()* function). Trimming the tree involves stepping through the tree in the order in which nodes were added and marking as invalid all child nodes whose parent nodes are invalid. This effectively breaks off branches where they directly collide with new obstacles and removes all nodes on these branches. Once the tree has been trimmed, it can be grown out to the goal. This can be performed in exactly the same manner as the proposed *RRTBiasedLimCon* algorithm used for the initial growing of the tree (Figure 3.24). However, depending on how the C-space has changed,

it may be more efficient to focus the growth towards areas that have been affected by adding biasing probability for sampling random points as the goal. To enhance the efficiency of the DRRT, it is recommended to use low-biasing probability in the planning phase. This makes the RRT covers the whole C-space and provides more branches to grow from when doing re-planning. While for the re-planning phase, high-biasing probability is better to minimize the processing time by directing re-growth towards the goal configuration.

Figure 3.26 shows a capture of a tree for a 2 DoFs circular robot that was trimmed and re-grown based on dynamic environment updates. The black and the blue cross in the figures represent the current configuration and the goal configuration of the robot in the C-space, respectively. Figure 3.26 (a) shows the initial tree that considers only static obstacles. Figure 3.26 (b) shows the tree trimmed and re-grown after detecting dynamic obstacle at the current robot location. Blue branches represent the original tree generated in the initial planning phase. Red branches are the newly re-grown branches by DRRT in the re-planning phase. Red boxes are invalid sampled nodes in static obstacles and green boxes are in dynamic obstacles. It should be noted that as the tree is re-grown, additional invalid nodes may be detected in static obstacles.

TrimRRT ()

1. $S = \emptyset$; $i = 1$;
2. while ($i < T$.size())
3. $q_i = T$.node(i); q_p = Parent(q_i);
4. if(q_p.flag = invalid)
5. q_i.flag = invalid;
6. if(q_i.flag $\neq$ invalid)
7. $S = S \cup \{q_i\}$;
8. $i = i + 1$;
9. T = **CreateTreeFromNodes**(S);

InvalidateNodes(obstacle)

10. E = **FindAffectedEdges**(obstacle);
11. for each edge $e \in E$
12. q_e = **ChildEndpointNode**(e);
13. q_e.flag = invalid;

RegrowRRTBiasedLimCon()

14. **TrimRRT**();
25. **GrowRRTBiasedLimCon**();

Main()

16. $q_{robot} = q_{init}$; $q_{init} = q_{goal}$; $q_{goal} = q_{robot}$;
17. **InitRRT**();
18. **GrowRRTBiasedLimCon**();
19. while ($q_{goal} \neq q_{init}$)
20. q_{goal} = Parent(q_{goal});
21. Move to q_{goal} and check for new obstacles;
22. if any new obstacles are observed
23. for each new obstacle o
24. **InvalidateNodes**(o);
25. if solution path contains an invalid node
26. **RegrowRRTBiasedLimCon**();

Figure 3.25: Pseudo code for DRRT algorithm (Adapted from Ferguson, 2006)

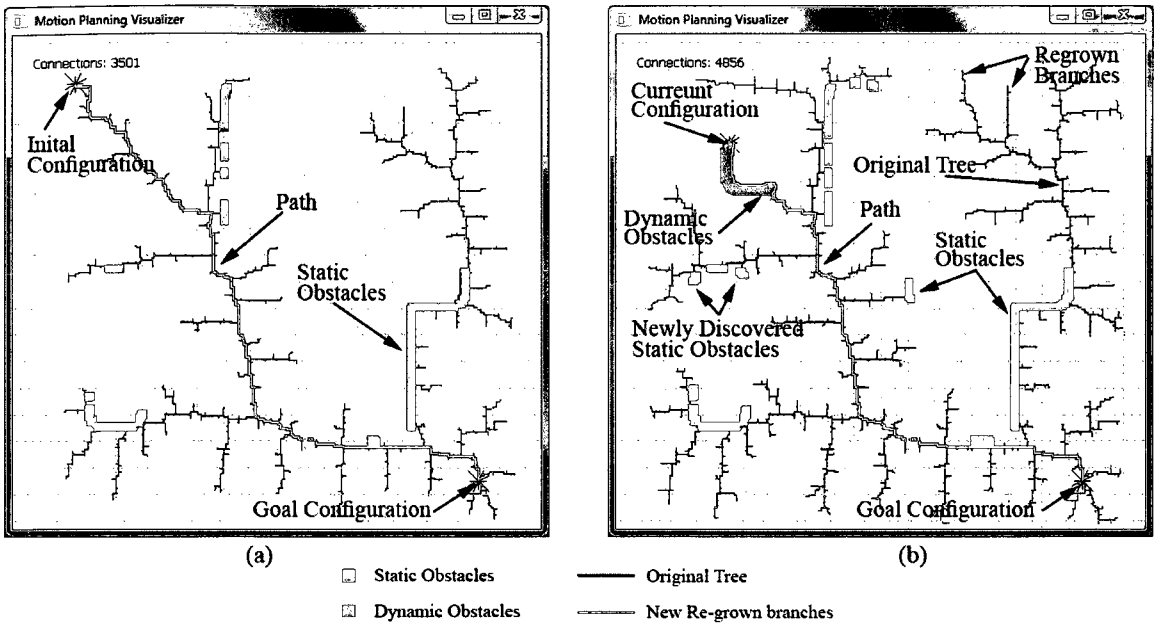


Figure 3.26: A capture of tree that is trimmed and re-grown based on dynamic environment updates: (a) Initial planning; (b) Re-planning after detecting dynamic obstacle

The proposed algorithm can be applied in a prioritized approach for efficient multi-equipment motion planning as explained in Section 2.6.4. Using this approach, each of the cranes is assigned a priority. Next, the cranes are picked in order of decreasing priority. For each picked crane a path is planned, avoiding collisions with the static obstacles as well as the previously picked cranes, which are considered as dynamic obstacles. This approach will be tested with two different case studies as will be shown in Section 4.6.

3.7 VISUALIZATION

To achieve better understanding of planning and re-planning processes, 4D simulations of the results have to be visualized while considering real-time rendering and interactivity. For real-time rendering, computer graphics has specialized APIs for 3D real-time rendering (e.g. DirectX or OpenGL) that can be used to do real-time rendering of the construction site. Interactivity is achieved by providing the user the ability to control and update the environment during executing the plan in the simulated scene. This could be achieved by integrating the framework into an interactive 3D system. The proposed interactive system should have the ability to capture real-time data from suitable sensing hardware that is accurate and synchronized with the real event timing (i.e. no delays).

While rendering the results, the generated path in the C-space should be used to calculate key frames with their interpolation parameters for determining the spatial positions each part of a hydraulic crane in the workspace as shown in Figure 3.27.

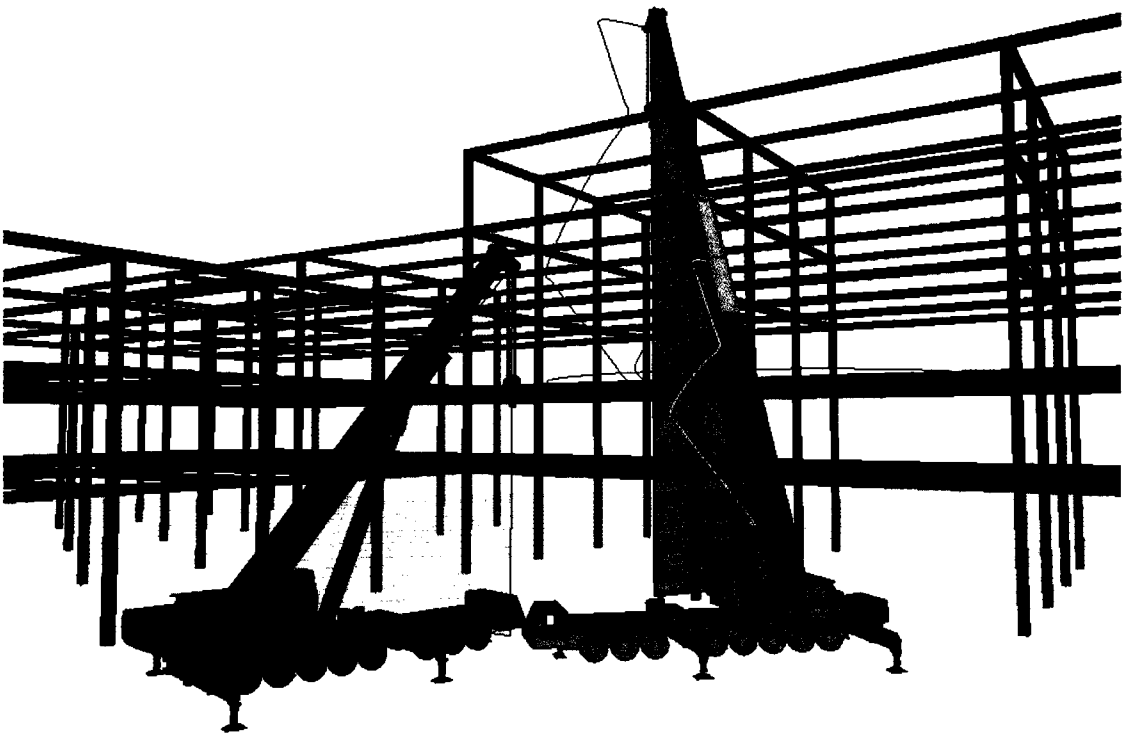


Figure 3.27: Paths computed for a hydraulic crane and visualized in the work space

Having accurate and detailed models for visualization can negatively affect the collision detection computation time, because more details means more triangles and more time to perform collision detection queries. To enhance the quality of the visualized models while maintaining optimized collision detection calculations, a well-known technique in computer graphics can be used where low-resolution proxy shapes are generated from the high-resolution models. Usually these low-resolution proxy shapes are bounding boxes that are attached to each element's center. Figure 3.28 shows the original crane side by side with a low-resolution proxy model that is composed from bounding boxes.

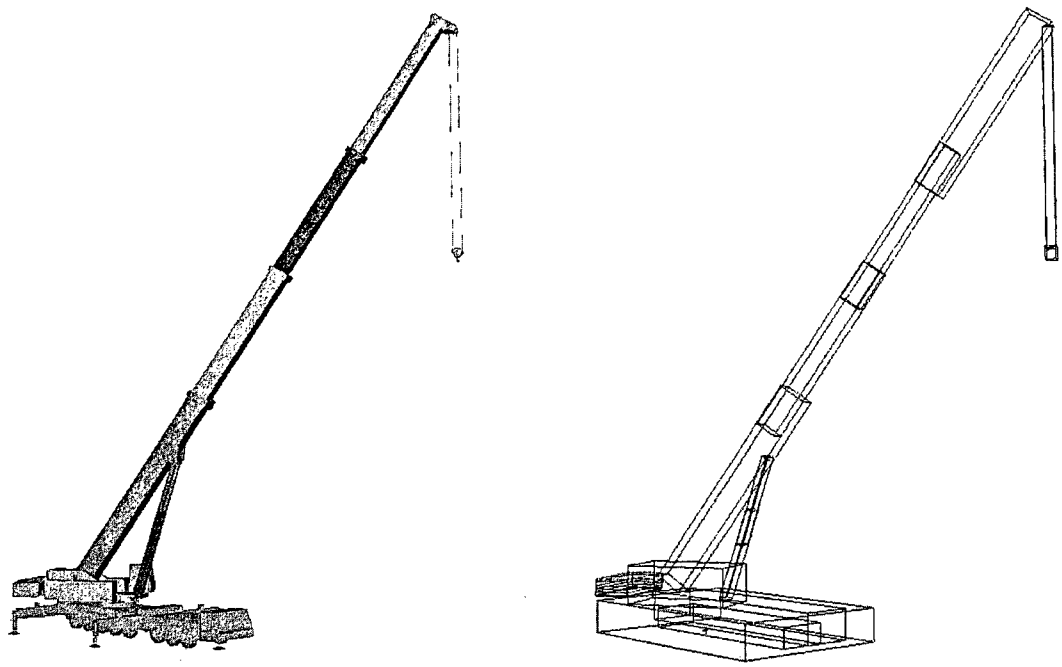


Figure 3.28: Two versions of the same crane: (a) 3D model of the hydraulic crane, (b) Bounding-boxes for collision detection

During the execution of the simulation in real-time rendering system, multi-sampling collision checking is required to detect obstacles that are smaller than the step between two successive nodes. Otherwise, the re-planning algorithm may not be able to detect small objects that are moving in the construction site.

3.8 SUMMARY AND CONCLUSIONS

The proposed framework for construction equipment motion planning and re-planning was discussed in this chapter. The framework focused on safety motion planning by considering engineering factors, realistic definition of the problem, and rapid re-planning

by proposing an efficient re-planning algorithm. This framework extends the previous research of crane path planning by: (1) Investigating several computational issues for defining the construction equipment, generating the C-space and planning/re-planning with the generality to be applied in the future for most heavy construction equipment; (2) Introducing several methods for satisfying safety requirements when applied to heavy construction equipment. These methods include the consideration of engineering constraints, rules of actions, obstacles dilation and critical volumes for avoiding unsafe paths; (3) Considering the dynamic properties of construction sites efficiently by proposing a re-planning algorithm which is able to provide information for operators in near real time to assist them in manipulating equipments more efficiently and safely; (4) Considering multi-equipment planning by integrating the re-planning algorithm in a prioritized motion planning approach; and (5) Proposing 3D visualization and simulation of the planning results while considering real-time rendering and interactivity to cope with the re-planning tasks. This framework makes the first attempt to integrate accurate path-planning and re-planning for heavy construction equipment in an interactive 4D simulated environment that has the flexibility to be updated using near real-time data captured by suitable sensors.

CHAPTER 4 IMPLEMENTATION AND CASE STUDY

4.1 INTRODUCTION

The work in this chapter introduces a prototype system for motion planning construction equipment. This system is referred to as ICE-Planner, which stands for "Intelligent Construction Equipment motion Planner." This system is able to solve off-line planning problems in addition to real-time re-planning that is required for multi-equipment planning and dynamic changing environments. The algorithms that were implemented in this work depend on RRTs for solving problems with high DoFs in continuous space. For re-planning, DRRT extension was investigated for making it suitable and efficient for re-planning tasks. To improve the performance of the DRRT and to make it suitable for path re-planning of construction equipments, the proposed modifications in Section 3.6.2 are implemented in this prototype. All algorithms that are implemented in this project are tested and evaluated for a hydraulic crane and a tower crane cases based on holonomic robot definition.

This chapter is arranged as the following: Section 4.2 introduces the system through a fast walkthrough over each of its components. Section 4.3 lists the development tools and technologies that are used to implement the prototype in one integrated system. Section 4.4 demonstrates the workflow of the system. Section 4.5 covers the implementation details in general and focuses more on the modifications and extensions that were made on the reviewed algorithms. Section 4.6 presents comparison results and evaluations for

the implemented algorithms through the different tests that were made with the system. Finally, Section 4.7 presents summery and conclusions of the proposed system.

4.2 SYSTEM COMPONENTS

The introduced prototype is composed of several components. Each of these components is responsible for a specific task, which all together provides a flexible system for modeling, solving and visualizing real-time motion planning problems interactively. Figure 4.1 shows the different components that form the system in addition to their relationships. The components in the figure are separated into three blocks based on their tasks. The first one is responsible for defining the motion planning problem using the modeling tools found in the 3D software package. The middle block is the main part for solving both the planning and re-planning problems based on the modeled environment in the first block. The last block is responsible for visualizing the results using the rendering capabilities in the 3D software for this task.

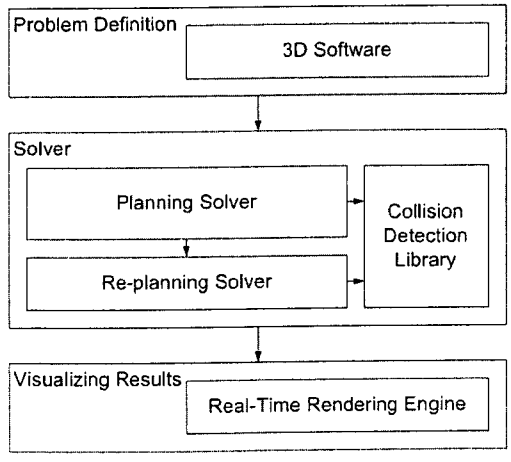


Figure 4.1: Main system components and their relationships

Actually these components are not explicitly separated but In fact, they are interwoven in different tasks to cope with the nature of real-time planning, where it is required to redefine the environment, solve and render iteratively.

4.3 SELECTION OF DEVELOPMENT TOOLS

To save effort and time, certain libraries and a 3D software package are adopted to fit with the proposed systems. Autodesk® Softimage® (formerly SOFTIMAGE|XSI) (Softimage, 2009) is a complete 3D software for visual effects and game production. Softimage is proposed here in the system as the main environment for creating and defining the motion planning problem in addition to interacting and rendering the results in real-time using OpenGL (OpenGL, 2009) and Direct3D (DirectX, 2009) APIs. Other components are integrated into it using its Software Development Kit (SDK) and its C++ API to ensure a seamless integration that takes full advantage of its 3D capabilities. The approach of defining the motion planning problem in Softimage will be described in Section 4.4.

In the solver component, a C++ library called the Motion Strategy Library (MSL) (Motion Strategy Library, 2009) was used for solving planning queries in the system. This library provides a wide range of randomized motion planning algorithms (e.g. RRTs). For this prototype, the library has been modified and extended to fit with the proposed system. Modifications are mainly for updating the code to be compatible and compiled with the Softimage C++ API. The library extensions are about adding new classes for interacting with the data in Softimage. This interaction is required in order to

read the motion planning problem directly from the Softimage scene which includes: kinematic properties and geometrical representation of the equipment in addition to the static and dynamic obstacles. More classes are added to the library for outputting the planning results as a 4D simulation in Softimage and utilizing the OpenGL API to visualize the results in real-time. Additional extensions to the library are made to create a new planning algorithm based on the proposed enhancements in Section 3.6.2. The implementation of the extensions will be introduced in Section 4.5.3.

Along with MSL, the Proximity Query Package (PQP) (PQP, 2009) was used for performing collision detection queries on obstacles found in the environment. Again this library has been modified and extended to be integrated seamlessly in the proposed system, in addition to the code modifications for compatibility issues, additional classes are implemented to be able to access the tessellated geometry directly from the Softimage scene at every simulation step. In the proposed systems, each simulation step is defined as one frame, and it is assumed that each second is composed of 30 frames. This assumption provides enough accuracy for the PQP library to detect unknown or dynamic obstacles in a construction site.

The re-planning solver was developed from scratch based on the DRRT algorithm with the proposed modifications and enhancements introduced in Section 3.6.2. The re-planning solver depends also on the PQP library to perform collision checks.

As mentioned, the implementation is done with C++ programming language. C++ is a general purpose programming language with a bias towards systems programming that is a better C language, supports data abstraction, supports object-oriented programming, and supports generic programming (Stroustrup, 1997). The amount of heavy computations required in the proposed system makes C++ stands in terms of execution speed in addition to program footprint and memory usage. C++ efficiency has been a major design goal from the beginning where it adapts the principle of "zero overhead" for any feature that is not used in a program. It has been a guiding principle from the earliest days of C++ that "you don't pay for what you don't use" (Abrahams et al., 2003).

For debugging the motion planning algorithms and visualizing the effect of their parameters on the way the C-space is sampled, algorithms were prototyped first using a special development environment in Softimage called the Interactive Creative Environment (ICE). It is a visual programming system for procedurally controlling simulations and deformations (Softimage, 2009). Using Softimage ICE, the tree structure of the RRT can be visualized in a 3D view, where the RRT nodes are represented as 3D particles in space as shown in Figure 4.2. Appendix A shows the full programming tree developed for visualizing RRTs.

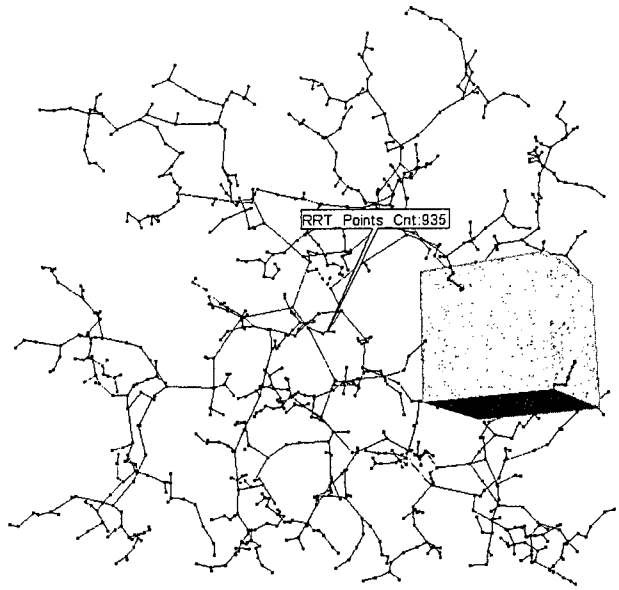


Figure 4.2: RRT visualized in 3D space developed by Softimage ICE

Having a visual representation of the tree structure is essential to understand how the algorithm is behaving based on the proposed enhancements and the defined parameters. Results from the visual experience are then utilized in the actual system that is implemented in C++.

4.4 WORKFLOW

In this section the general workflow of the system is explained. It is divided into subsections based on the general steps required to define, solve and visualize a hydraulic crane motion planning problem as our main case study. The subsections are organized as the following: Subsection 4.4.1 covers the essential tasks for defining the crane. Subsection 4.4.2 explains the workflow for defining engineering constraints for the hydraulic crane. Subsections 4.4.3 and 4.4.4 clarify the difference between static and

dynamic obstacles by explaining what type of objects in the construction site should be considered static or dynamic in the cases of single and multi-equipment planning. Subsection 4.4.5 explains how to define the initial and goal configurations to setup a planning query for the crane. Subsection 4.4.6 explains how to execute the solver and specify its parameters to compute and visualize the results properly. Subsection 4.4.7 discusses the re-planning phase and the level of interactivity provided by the system.

4.4.1 Crane Definition

The workflow of the system starts by defining the motion planning problem. This includes defining the geometric and kinematic model of the crane in addition to defining both the dynamic and static obstacles. As stated before, computer graphics and animation tools in the 3D software are utilized and customized to define different aspects of the motion planning problem as discussed in the following.

For the crane, basically it is defined through three main steps; the first step is to build the geometry that represents the crane's shape used in visualization and generate the bounding boxes used for collision detection; the second step is to define the kinematic properties and relationships between its components. The final step is to specify points on the crane that will be used to construct critical volumes.

The crane's geometry should meet a list of requirements in order to be defined correctly for the solver. These requirements are: (1) All controllable components of the 3D model should be extracted into separate objects. For the hydraulic crane, there should be at least 4 separate objects: truck, cabin, boom and cable; (2) Transformation axis for each part

should be positioned and oriented based on the DH-notation explained in Section 3.3; and (3) Visualization geometry should be composed of a polygonal mesh with clean topology and composed of the minimum amount of triangles in order to optimize real-time rendering.

In this system, constructing the crane geometry that fulfills the previous requirements is done with polygonal modeling tools available in the 3D software. This enables great flexibility for defining the shape of the crane with a high level of detail. Of course building highly detailed model of the crane is better visually but it causes higher geometry complexity that can slow the calculations, especially when performing collision detection queries on the mesh. To overcome this problem, the proposed technique in Section 3.7 of having bounding boxes is applied here.

Additional simplification method can be done for reducing the DoF as proposed in Section 3.3.1. Using this method, the results that are computed on the simplified version of the crane should be transferred to the high-resolution version in order to visualize the results on it. This can be achieved in a better way by establishing connections between the crane parts in the two versions. These connections are created in the 3D software using either mathematical expressions or motion constraints. Figure 4.3 shows the connection between the two versions, *Exp* indicates a mathematical expression relation between the two objects while other connections are motion constraints.

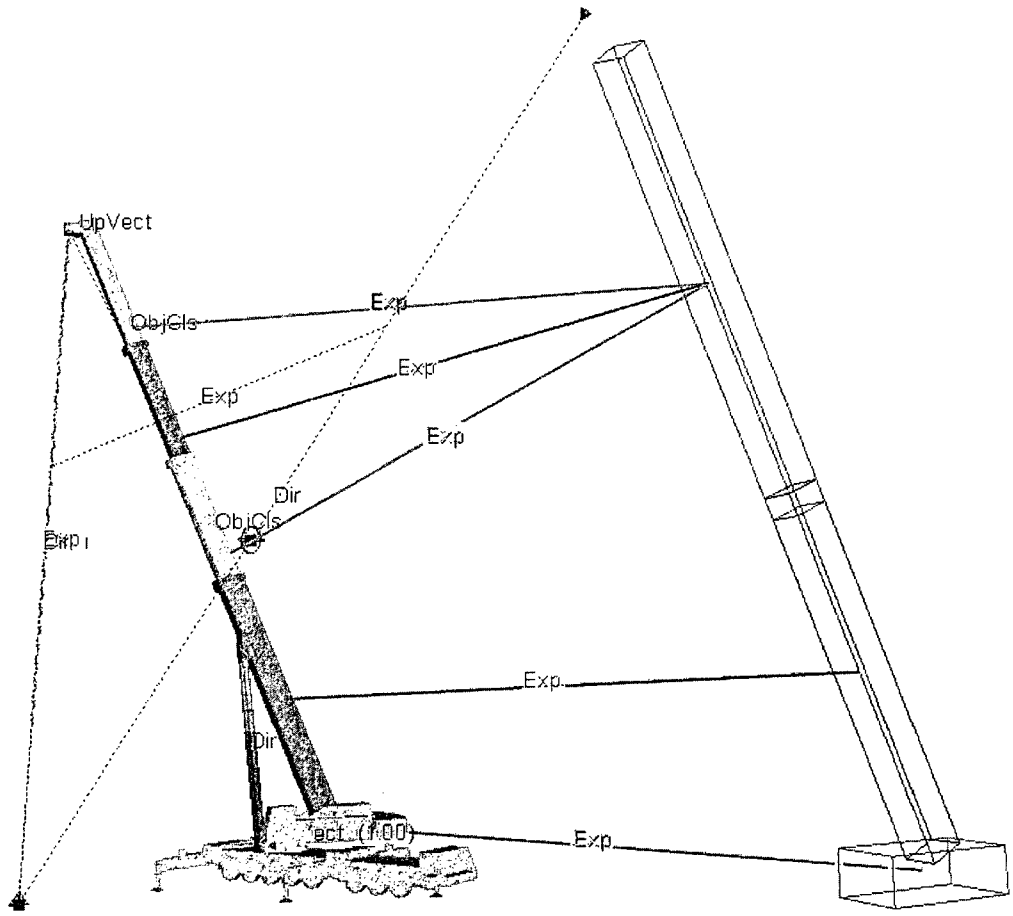


Figure 4.3: Relationships between the computational and the visualization version

In this case the results are visualized instantaneously on the detailed version while the solver is applying the results on the simplified one.

To apply this method on a hydraulic crane, it requires defining linear control equations between the four prismatic joints in the detailed representation and the single prismatic joint in the simplified representation. This can be done in the 3D software using the "*link with*" function. This function creates a direct relationship between two parameters where one parameter depends on the state of another parameter (Softimage, 2009). For the two

versions of the crane model, the translation of the four prismatic joints in the detailed model are linked to the translation of the boom in the simplified model as shown in Figure 4.4. The vertical axis represents the extension values from the simplified model, while the horizontal value is the extension value for the visualization model. The linking equations are defined based on the equation presented in Section 3.3.1 to provide the correct sequence for extending boom parts. As an example if the extension in the simplified model is 20m then based on the plotted equations, the extensions for the joints in the detailed model are: 0m, 0m and 6m starting from the tip joint respectively.

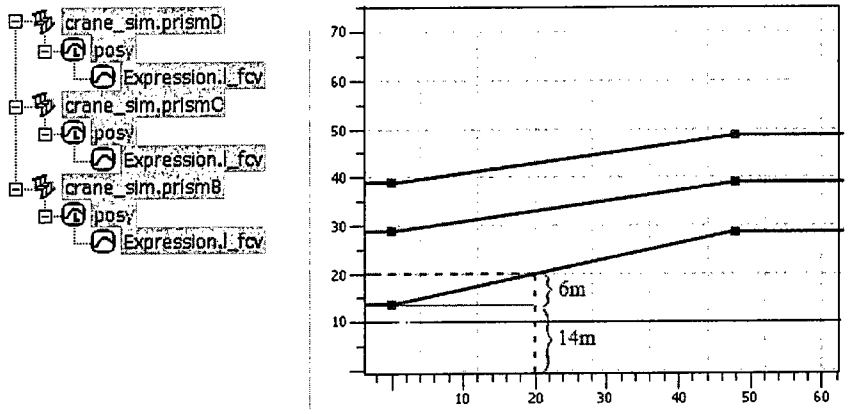


Figure 4.4: Plot of the function curves defined to drive the boom extension based on the computation results applied on the approximated model

Each part of the modeled mesh is separated into logical parts as mentioned in the requirements. Transformation axis are then assigned to them and arranged in a hierarchy that reflects the relationship between these parts. Figure 4.5 shows a schematic view from Softimage where the parts of the hydraulic crane are arranged logically. At the root is the cabin, it should be named: "*robot*" in order for the plug-in to detect it as the solvable

construction equipment. The first child of the cabin is the boom base, next is the boom extension part, while the last two nodes are for the cable.

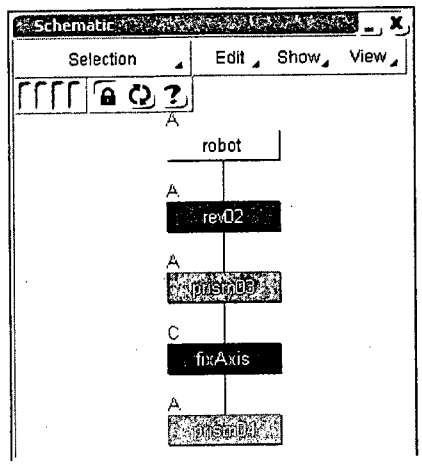


Figure 4.5: Schematic view for the hierarchy of a hydraulic crane

As noticed in Figure 4.5, the schematic view shows two nodes in the hierarchy of the hydraulic crane for the cable. The first node is a dummy object introduced to provide a world-oriented axis parent for the cable, which is represented with the last node. In order to make the dummy node provide a world-oriented axis to the cable, it should compensate the rotation of its parent node (Boom Extension). This can be done in the 3D software using motion constraints and expression functions available in the software. The idea is to add a new *null* object outside the hierarchy and make it follow the X-Z plane projection of the *fixAxis* node. This can be done by writing mathematical expressions that connects the X and Z parameters of the null to the X and Z of the *fixAxis*. After the *null* is attached to the projection of the *fixAxis*, the direction of the *fixAxis* can be constrained to the *null* using the *AddConstraint("Direction")* function in Softimage. Since the *null* is positioned at the projection of *fixAxis*, this constraint will make the *fixAxis* always have

matching orientation to the world axis regardless of its parent axis; thus the cable is then transformed relative to a world oriented parent.

This workflow of fixing the orientation of the cable's parent can be extended for implementing the physical behavior of the cable and considering it in the solver. As the boom is moving, the cable reacts correctly based on the movement of the boom and the physical properties of the cable. The differential equations for the cable dynamics can be implemented directly with this workflow as mathematical expressions on the projected null which drives the direction of cable based on the boom's movement. This feature is out of the scope of this research and it will be discussed in detail in future work Section 5.3.

For completing the equipment definition, the DoF types and their ranges should be defined in the kinematic properties for each component of the model. This is done by checking the correspondent axis under the DoF type (rotational or translational) and specifying its limits using the minimum and maximum parameters. Figure 4.6 shows the property page for one part of the crane's boom where it has a revolute DoF around its local Y axis with the limits of 0° to 90° degrees.

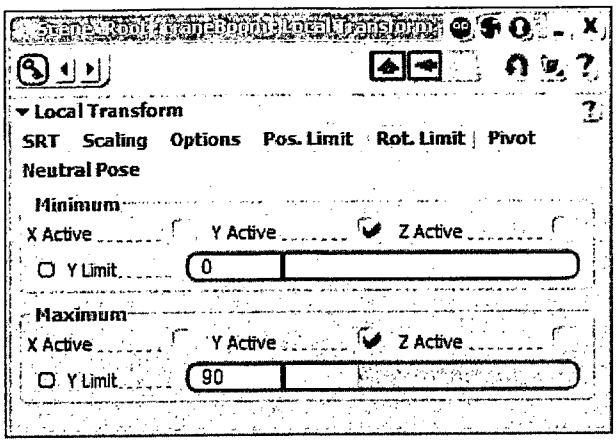


Figure 4.6: Local kinematics property page used to define the DoF and its limits

For the case study, joint limits are applied to each DoF of the hydraulic crane. First, the cabin has joint limits on the rotational Y-axis between -180° to 180°, this allows full 360° yaw for the crane boom. For the boom root, it has joint limits on the rotational X-axis. The joint limits for this DoF are computed dynamically while generating the path based on the load charts and the current crane configuration as will be explained in Section 4.5.2. For the boom extension, it has joint limits on its translational X-axis. Again the joint limits for this DoF are computed with respect to the load charts and current crane configuration.

In order to construct 3D surfaces that represent critical volumes, several points on the crane should be specified to encompass its critical volumes. For the hydraulic crane case, a critical volume bounded by its boom and the cable is defined by at least three points. These points are attached as shown in Section 3.5.2.

4.4.2 Engineering Constrains

In this prototyped system, engineering constraints are applied for hydraulic cranes based on the proposed approach in Section 3.4. For a hydraulic crane, engineering constraints are mainly derived from its load charts. The load chart that is used for the hydraulic crane (TMS870) is shown in Figure 4.7.

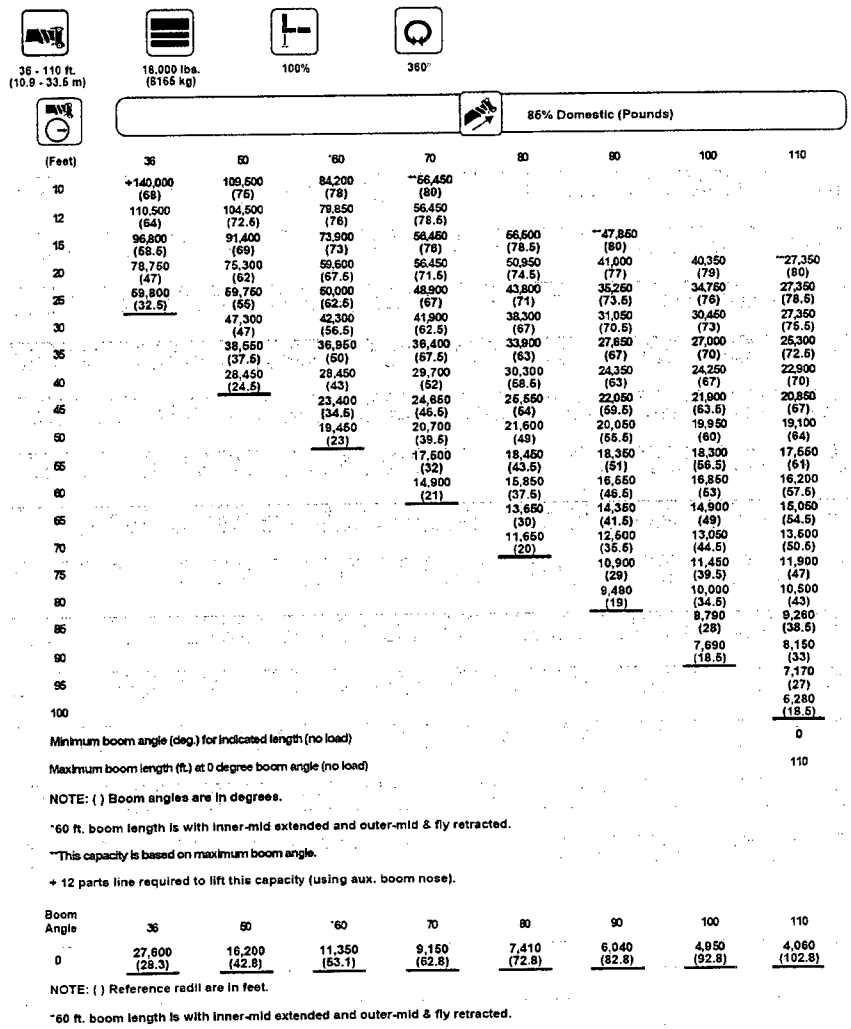


Figure 4.7: Load chart of a hydraulic crane (Groove Crane, 2006)

Based on the lifting task requirements and crane configuration, queries for the allowable boom extension and angle are searched from a database of the load chart. This problem is solved by a simplified searching algorithm that is presented in Section 3.4. Figure 4.8 shows the Graphical User Interface (GUI) for specifying the task and configuration parameters for a hydraulic crane:

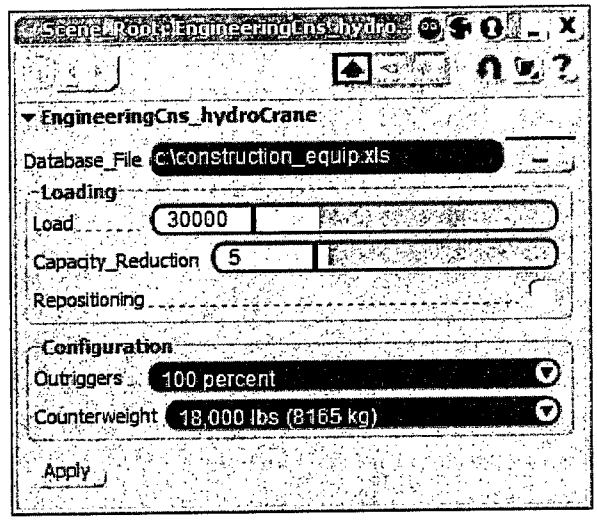


Figure 4.8: Hydraulic crane engineering constraints GUI in Softimage

The flowing explains the parameters in this dialog that are required to define the engineering constraints for a hydraulic crane:

Datanbase_File: this parameter specifies the MS Excel file that contains the load chart data for the hydraulic crane.

Load: a float value that represents the object's weight that is required to be lifted by the crane. For the case study this load is expressed in (lbs).

Capacity_Reduction: a percentage used to reduce the load chart capacities caused by improper crane setup; Appendix J demonstrates several reasons for capacity reduction of a hydraulic crane.

Repositioning: this option toggles the positioning mode which means that the current task is just about posing the crane without lifting any objects.

Outrigger: this option represent outriggers state, if they are extended for the current task or not.

Counterweight: specifies the value of the equivalent counterbalancing weight that balances a load.

4.4.3 Static Obstacles

Static obstacles are those obstacles that represent the initial information about static objects in the construction site that are known in advanced by the planner so they are considered during the planning phase. Examples of these obstacles include available buildings, electrical poles, etc that are already known from the Building Information Model (BIM) (Kumi et al., 1997). 3D models for these obstacles can be imported using capturing hardware (e.g. 3D laser scanners) or modeled directly based on engineering and architectural designs. In most cases these models may require extra processing to generate clean 3D models that can be used efficiency by the planning solver. 3D

polygonal modeling tools available in Softimage can be used in constructing new 3D models that represent static obstacles or cleaning imported data from capturing hardware.

In the proposed workflow, defining a 3D model as a static obstacle in Softimage is done by simply parenting them under a specific model with the name *obstacles*. The solver then considers all objects under this model as static obstacles and performs collision checking during the planning phase. Any type of geometry can be used to model a static obstacle object. However, if any motion information is attached to 3D objects that are specified as static obstacles, it will be ignored, and only a snapshot of the geometry from the first simulation step is taken into consideration by the solver.

Although static obstacles are not allowed to update during the planning phase, they are still adequate for representing the updates of the construction process. This is because the update rate needed for the construction processes is much less than the update rate of equipments tasks execution. Thus, updated 3D models of the building can be still considered as static obstacles by satisfying this assumption and updating the building models between successful equipment tasks. In case where some building components are updating while the equipment is executing its tasks, these obstacles should be defined as dynamic obstacles in order to be considered by the solver.

4.4.4 Dynamic Obstacles

Dynamic obstacles are those obstacles that their representation is updated or detected completely while executing the initial plan. In the case of multi-equipment motion planning, the high priority equipment is considered a dynamic obstacle for the less

priority one. With this definition, objects defined as dynamic obstacles have changes in their geometry because of their movement or simply because they have changes in their shape.

Dynamic obstacles in the construction site (e.g. workers and construction equipment) are represented using 3D objects with motion applied to them. This motion can be generated either: (1) interactively by allowing direct control of the obstacles through Softimage animation tools Scaling, Rotation, and Translation (SRT) for transforming objects separately or relatively in hierarchies; (2) Using the sensor capabilities of the equipment or real-time capturing hardware; or (3) Using the solver to generate motions computationally in the case of multi-equipment planning.

3D objects with dynamic shapes are used to represent critical volumes that are constructed to block the solver from generating paths into unsafe volumes in addition to generating components that are defined based on real-time capturing hardware. Such obstacles have initial representation and are their shape is updated using 3D capturing hardware. For the implementation part in the 3D software, critical volumes are defined using the concept of shape animation. Shape animation (sometimes called morphing, blend shapes, or endomorph), is the process of deforming an object over time. Deformation is done by animating geometrical points based on the captured poses (snapshots) of the object.

For cases of updating obstacles representation, real-time data about obstacles updates using sensors can be accessed inside Softimage using the device drivers. Device drivers are connections between a device plug-in (e.g. motion capture system or UWB sensors)

and Softimage that get and set the values or execute commands or events. These devices can be used to animate the objects based on captured information. For this research, dynamic obstacles are simulated by providing interactive control over their movement concurrently while the crane is moving inside the 3D environment.

4.4.5 Defining Initial and Goal Configurations

The next main step for defining the motion planning problem for a hydraulic crane is to define the initial and goal configurations of its task. The solver will try then to find a feasible path between these two configurations while considering environment obstacles and constraints.

Currently, the process for specifying these configurations is done by first posing the crane using the transformation tools in the 3D software, then marking all parameters on each joint that correspond to a controllable DoF, and finally adding key frames to the marked parameters. The solver searches on each part of the equipment for these key frames. It is assumed that the first key frames have the values of the DoFs at the initial configuration and the last key frames have these values for the goal configuration. Thus, for each DoF of the equipment, there should be at least two key frames to be able to specify the motion planning query. To make this process easier (especially for cases where it is possible to have multiple queries of the crane), it is preferable to create a marking set for the crane. This way all parameters are grouped into a set for easy key framing.

4.4.6 Planning and Visualization

The planning solver is executed once the *MotionPlan_RealTimeDRRT* command is called. This command is part of the implemented motion planning plug-in for Softimage and it requires specifying a list of parameters in its GUI as shown in Figure 4.9.

Actually, these parameters are all related to the RRT algorithm that is used for performing the planning and re-growing tasks. The following is a brief overview for these parameters:

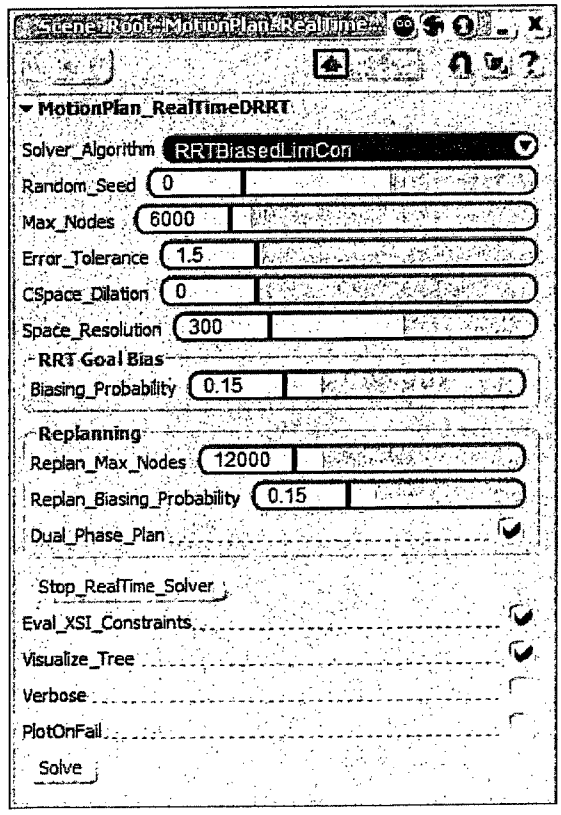


Figure 4.9: *MotionPlan_RealTimeDRRT* GUI

Solver_Algorithm: Specifies the RRT extension that will be used to search for the feasible path. Only extensions that depend on growing single tree are available. This is required because the data structure in the planning phase will be used in the re-planning phase and the proposed re-planning algorithm is restricted to single tree in order be able to re-grow it again.

Random_Seed: An integer seed number for generating the uniform random distributed values when sampling new random configuration in the RRT. Changing this value affects the detailed structure of the tree while the general structure of exploring the C-space uniformly will not be affected. Thus, this value should not be changed in order to be able to generate the same tree structure when solving the same problem again.

Max_Nodes: The maximum allowed number of nodes that the RRT can generate while solving the problem. If this number is reached before finding a solution, then the solver will return failure.

Error_Tolerance: The solver considers that the crane has reached the goal if the distance in the C-space between the crane configuration and the goal configuration is less than a threshold value. This threshold value is computed in the workspace for each DoF by multiplying the *Error_Tolerance* with the speed factor for each DoF that is calculated in the metric (Section 3.6.1).

CSpace_Dilation: This is a float value that specifies the amount of uniform expansion of the obstacles in order to avoid semi-free paths.

Space_Resolution: Is an integer number that determines the number of subdivisions along any dimension of the hypercube that is used to calculate the metric coefficients. These coefficients are used also to control the movement speed for each DoF in the simulation, thus the *Space_Resolution* parameter affect the overall speed of the simulation.

Biasing_Probability: Is the probability value for biasing the random node generation towards the goal configuration instead of uniform random generation. High probability values direct the tree generation towards the goal configuration instead of uniformly capturing the C-space.

Replan_Max_Nodes: The same for the *Max_Nodes* parameter but this is for the re-planning phase instead.

Replan_Biasing_Probability: The same for the *Biasing_Probability* parameter but this is for the re-planning phase instead.

Dual_Phase_Plan: Activating this parameter allows the solver to do re-planning in case the solver failed to find a solution in the planning phase. This feature is more efficient when solving complex problems, because it utilizes the previously generated tree which can be re-grown using a different biasing probability to find the solution.

Eval_XSI_Constraints: This is a key parameter that affects updating the equipment configuration after each simulation step. It allows the solver to either consider all constraints, mathematical expressions and simulation equations applied on the equipment in Softimage or just to depend on internal transformation calculations presented in the DH-notation to determine the equipment configuration. Internal computations are much faster and efficient; this is because all computations are made directly on available transformation matrices. Evaluating Softimage constraints enables the solver to consider them while generating the plan which allows solving complex problems with any physical simulation constraints applied to it (e.g. swinging simulation of the crane's cable based on the weight inertia and wind effect).

Visualize_Tree: This option allows visualizing the generated RRT in a separate window for further analysis and debugging.

Verbose: This is another debugging option for logging computational details to the script editor. Appendix H shows sample of the verbose logging for the solving process.

PlotOnFail: In case the planner returns failure, this option plots all successful nodes in the tree as keyframes on each DoF. Figure 4.10 shows the plotted points for the first DoF of the hydraulic crane (swing angle). Points start spreading from the initial value of the DoF (55°) all over the specified range of the DoF (-170°,170°) as the tree expands in the C-space. The shaded area represents the C-Space occupied by obstacles. As can be noted, valid nodes do not exist in this area.

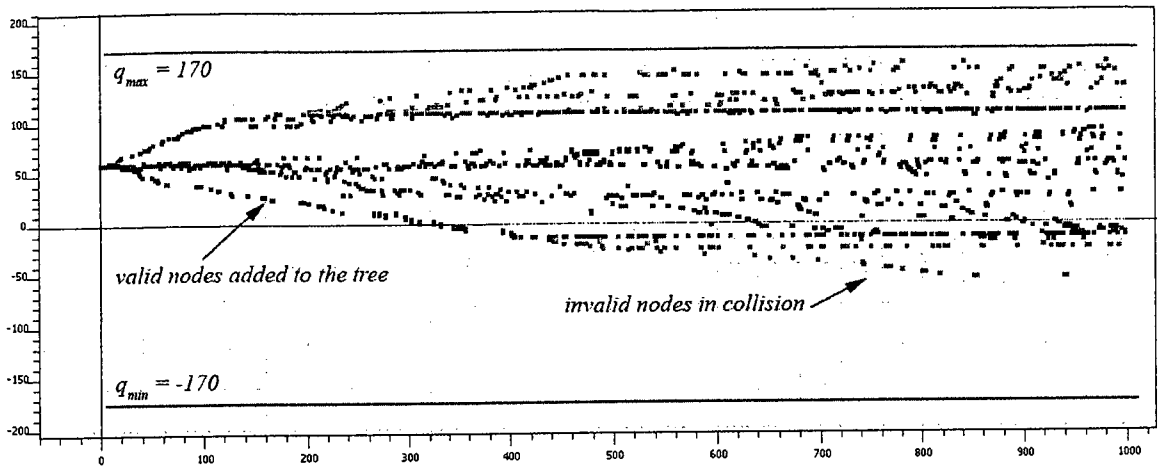


Figure 4.10: Plot of the RRT for the first DoF of the hydraulic crane

After executing the command and successfully finding a feasible path, the path is stored along with the full tree information in memory. For visualization, the 3D software reads the path information from memory to move the crane over the feasible path that was found by the specified algorithm. Of course, the feasible path at this level is only based on static obstacles. Dynamic obstacles are considered later in the re-planning phase.

In case the planning phase fails (the allowed number of nodes is reached before finding a solution), the last updated RRT is cached and a new planning query is then treated as a re-planning problem where the existing tree is trimmed and re-grown (this is only if the *dual_phase_plan* parameter is activated).

In case the re-planning phase fails also, the command outputs all random configurations that are added to the tree until it failed on the 3D model. This can help in fine tuning the parameters (e.g. biasing probability) based on the visual feedback of how the solver was progressing from the recorded tree.

4.4.7 Re-Planning and Interactivity

When the planning phase succeeds, the implemented plug-in executes collision detection for each simulation step while the crane is moving over the initial path. Based on the specified amount of the simulation step, a collision detection query is computed for the next equipment step. If a collision is detected, event-based command is invoked to perform re-planning task in order to fix the initial path based on the newly detected obstacles.

This event system enables a high level of interactivity where it is possible to add any type of motion to the obstacles while the crane is executing the initial path. For example, it is possible to simulate a truck moving around while the crane is moving and see its response directly. Figure 4.11 shows snapshots for the example where an excavator model is moved near the crane while it is moving along the planned path. The excavator is detected as dynamic obstacle at simulation step 25 where the re-planning phase is executed to update the path and avoid the detected obstacle.

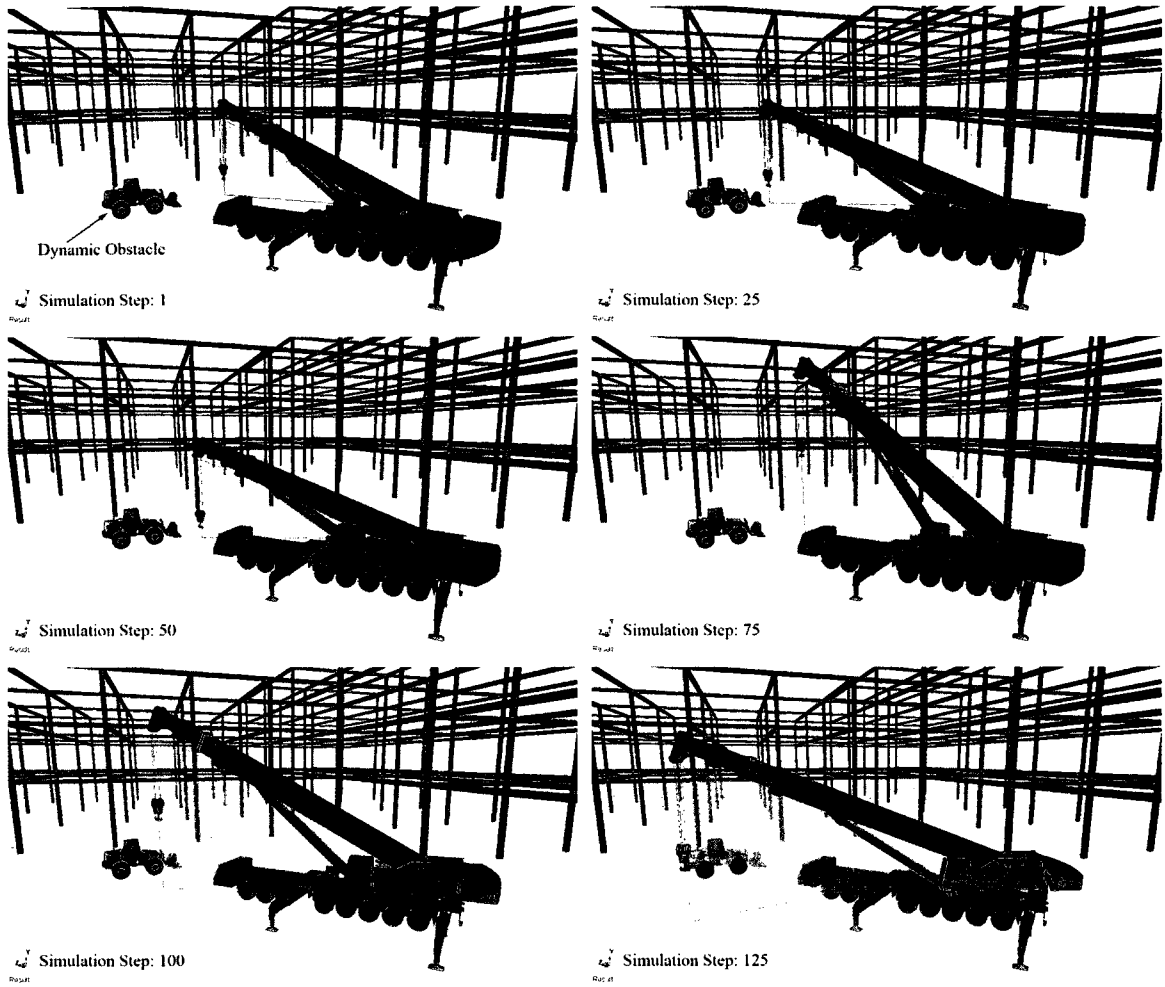


Figure 4.11: Excavator model moved near the crane and detected as dynamic obstacle while the crane is executing the path

4.5 IMPLEMENTATION DETAILS

To simplify the development process and reduce the complexity of the ICE-Planner, it is divided into several components that interact together in certain logic to provide a fully integrated system to model, solve and visualize construction equipments motion planning in real time. Figure 4.12 summarizes the main components of the ICE-Planner and shows the relationships between them.

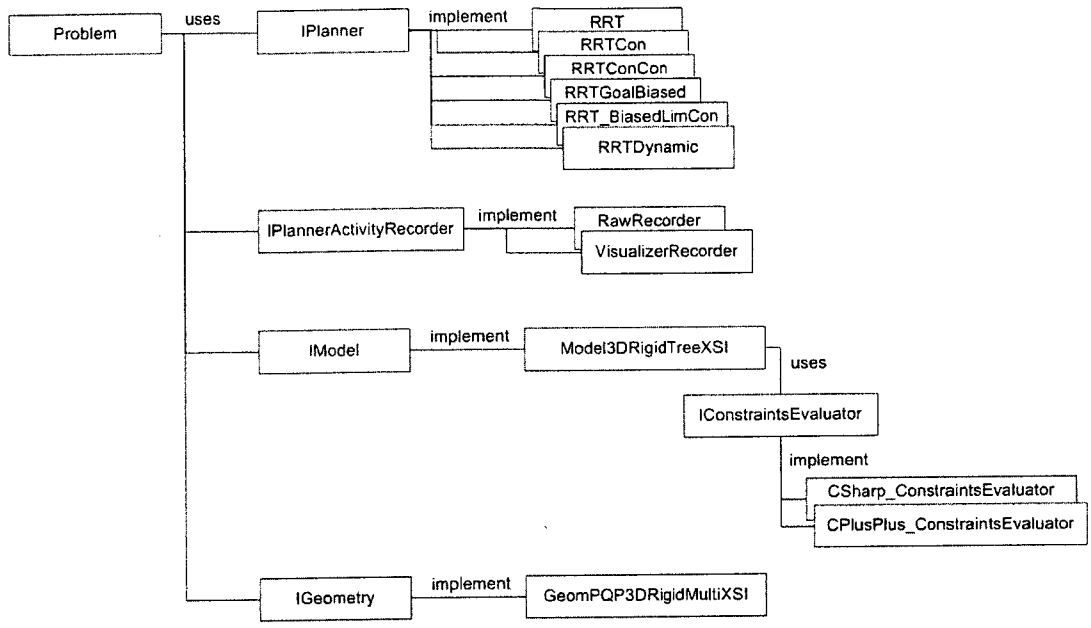


Figure 4.12: Main components of the ICE-Planner

The "uses" relationship calls directly onto the member object and uses its services, while the "implement" relationship is a specialization of its parent type. The *Problem* component is the container for the different aspects of the motion planning problem. The *IPlanner* is an abstract interface that is implemented by several classes each representing a specific algorithm. This interface unifies the usage of different algorithms supported by the system. The *IPlannerActivityRecorder* is the interface that implements the two classes (*RawRecorder* and VisualizerRecorder). *RawRecorder* class plots the nodes from the generated RRT on the equipment 3D model as key frames for each DoF of the path in case of success or as tested configurations in case of failure. The *VisualizerRecorder* class renders the RRT nodes in the C-space for the first three DoF of the equipment. The *IModel* implements the class that has the metric and other functions related to defining and accessing the kinematic structure of the equipment. The *IConstraintsEvaluator*

implements classes for validating sampled configurations against engineering constraints using the proposed LVA in Section 3.4. The *IGeometry* implements class for performing collision detection calculations using the PQP library.

Figure 4.13 shows the lifetime of various objects involved in the operation of planning phase. The blocks laid horizontally list the main objects in the system while the vertical axis represent the lifetime of each of these objects.

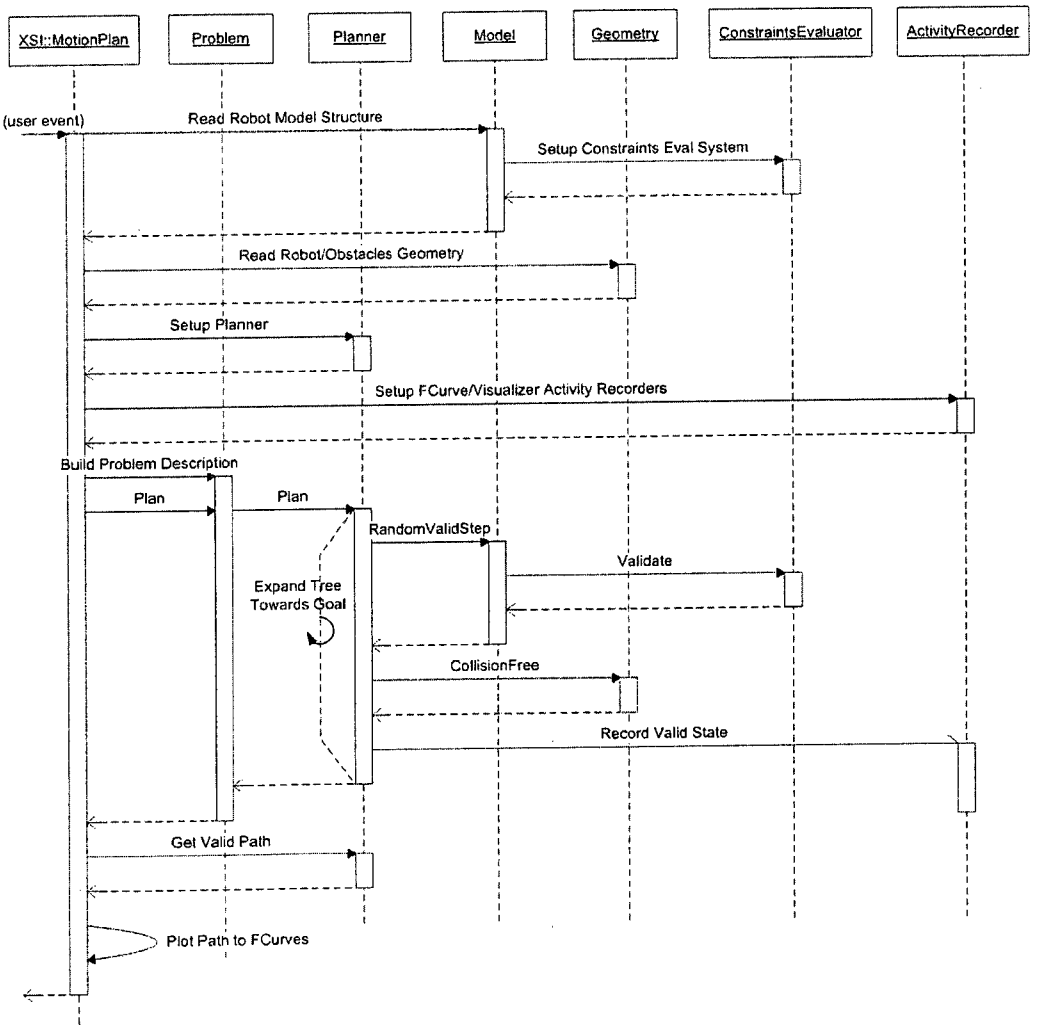


Figure 4.13: Sequence diagram of the ICE-Planner

4.5.1 Problem Modeling

The major implementation effort for the problem modeling task is to be able to read the problem data directly from the 3D software. Problem data that needs to be retrieved consists of: (1) geometry that represents the crane and the obstacles shapes, (2) kinematic properties for all crane parts, and (3) initial and goal configurations.

To read the geometry from Softimage, the geometry first needs to be tessellated in order to perform collision detection calculation using the PQP library. Fortunately, Softimage provides a method for accessing tessellated meshes directly without the need to implement this separately.

Before using the tessellated geometry data in the collision detection library, it is first passed to another class for further processing. This class is responsible for extending the obstacles uniformly with a specific value to avoid semi-free paths around the original obstacles. Since dilation is done as a uniform extension around obstacles, it is possible to utilize a simplified approach for dilation. This approach depends on extending the obstacles geometry based on its vertices normals. This is done by first accessing the normal vectors on each point of the obstacle, and then each point is translated along its normal vector based on the specified value. Figure 4.14 shows the previous process for an I-beam.

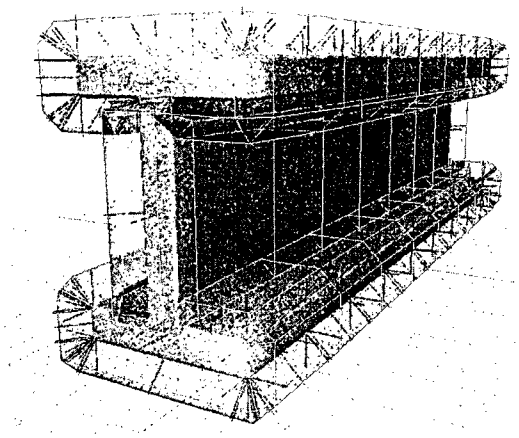


Figure 4.14: I-beam dilated along its points normals

In addition to accessing geometry defined in the scene, geometry for critical volumes needs to be generated and defined as dynamic obstacles for the solver. In the case of a hydraulic crane, the points that are attached to the crane are only changing their positions based on the configuration of the crane, while their number is always the same. This fact can be taken into account to enhance the performance of generating and updating the 3D surface. Thus, instead of regenerating the geometry from the points at each simulation step to reflect the updates on the 3D surface, the geometry generation can be done only in the first simulation step, then as the configuration of the crane updates, points positions of the constructed 3D surface are moved to update the 3D surface shape. This simplification can enhance the calculation time for cases with complex shapes.

The way the kinematic properties of the equipment are defined in the implemented system is different from how it was defined in MSL. In MSL's hierarchical equipment model, DoFs are defined based on the DH-notation. While in our implementation it is defined using the local affine transformation matrix that is formed from six parameters.

This modification is made in order to make it compatible with the kinematic properties defined in Softimage. As a result, accessing the kinematic properties is straightforward. Softimage provides direct access to these properties through the object model in the SDK.

Having a compatible matrix representation with Softimage is important to boost the performance when evaluating dynamic constraints and mathematical expressions applied to the equipment. This is because the solver has to apply these transformation matrices to the equipment model before each simulation step in order to evaluate any applied constraints based on the computed configurations.

Also in order to increase the flexibility for defining equipment hierarchies, a modified structure is implemented that is able to support articulated equipments with branching hierarchies without any limits on the number of links. This generalized structure should be capable to model most type of equipment easily without the need to readopt the system for it specifically. Thus, it is straightforward to solve other types of equipment that have different number, ranges and types of DoFs. In this research, in addition to solving hydraulic cranes, the developed prototype system is used to solve tower cranes that have totally different kinematic structure from hydraulic cranes. By following the same workflow that is explained in Chapter 3 for hydraulic cranes, another case study for tower cranes is solved and presented in Section 4.6.

Reading the initial and goal configurations is done by looping over all equipment parts and retrieving the parameter values at the first and last key frames applied to these parts.

Any key frames in-between these two key frames are ignored and will be overwritten by the solver's results.

4.5.2 Engineering constraints

As presented in Section 3.4, engineering constraints are considered by developing a separate engineering agent that collaborates with the solver to generate safe paths. In order for the engineering agent to make decisions, it requires two main inputs:

1. Task parameters defined by the user that include in the case of a hydraulic crane: lifted load, capacity reduction percentage, outriggers state, and counterweight.
2. Current configuration which is the current pose of the equipment that is computed by the solver in the last simulation step. From this configuration, the solver states a query about the current values of all DoFs that affect the allowable ranges of the DoF that is going to be solved in the next simulation step.

For implementation, engineering constraints data (i.e. load charts) are organized in tables and stored as a MS Excel file. Data stored in the file should follow specific organization in order to be referred correctly by the system. In Softimage, another separate plug-in is developed that is responsible of: (1) retrieving user inputs regarding the task parameters, (2) accessing the data in the MS Excel file based on the task parameters, and finally (3) generating the logic and datasets that are customized based on the task parameters as an engineering agent and attaching it to the crane model.

Accessing the data in the MS Excel file is done based on the specified parameters of the crane task. These parameters define the data cluster that satisfies crane task. To simplify this process, data is organized in a logical way to make it easy to define data clusters as partial tables of the spreadsheet.

The logic that is included in the engineering agent is based on the search algorithm that is presented in Section 3.4. This logic in addition to the datasets from the load charts are all generated based on the task parameters (i.e. counterweights, load, state of the outriggers) and integrated into an abstract customized agent that is attached to the 3D model of the crane. Attaching the engineering agent is done by adding the programming code that includes the searching algorithm and the datasets to the equipment model as a string property. The content of this property is then executed internally by the solver on demand where the solver is able to interpret/compile the code that is attached to the equipment model and execute it as an attached library. This technique was first presented by (Bahnassi, 2009) for creating new scripting structure for 3D models where they have their own logic and algorithms attached to them. The original concept supports attaching code for scripting languages (e.g. Jscript and VBScript). Unfortunately, using scripting languages to implement the algorithm for searching the engineering constraints is not acceptable in terms of performance for real-time planning. This is because the evaluation time of a script is so much longer compared to compiled code. To overcome this performance issue, an extension is developed to support C# (or C++) that is compiled and called by the solver when planning. The performance of this extension is enhanced in the

ratio of 250 times from using scripting languages. Appendix G lists the auto-generated C# code for the engineering agent for a hydraulic crane.

4.5.3 Planning/Re-Planning Solver

As mentioned before, the implementation for the planning solver is derived from the MSL API. To be able to integrate it with Softimage, the library was modified and compiled with MS Visual Studio 2008. Algorithm extensions proposed in Section 3.6.2 are implemented as new classes in MSL. First, the proposed metric that is introduced in Section 3.6.1 is implemented in a separate class where it has members defined dynamically based on the problem definition. These members include DoF ranges, speed and weight coefficients for each dimension in the C-space. This class includes several key functions. These are: (1) the distance function for calculating the distance between two states in the C-space, (2) interpolation function for extending incrementally towards sampled node, (3) and a difference function for subtracting two states in the C-space. The difference function is used to compare if the difference between the current node and the goal node is less than the allowed error tolerance in which case the goal will be considered reached and the tree returns a successful path. Appendix D lists the C++ code for the main part of the implemented class.

Enhancements and modifications related to the random node generation and to the tree extension are implemented in a new RRT class. In this new RRT class, the random node generation supports goal biasing based on predefined probability as discussed in Section 3.6.2. For the tree extension, an additional class is implemented for the modified Connect

heuristic proposed in Section 3.6.2. It supports the engineering rules of actions and it has limited greedy behavior that enhances the optimality of the path. Appendix E lists the source code in C++ for the implemented algorithm.

For the re-planning, a framework is implemented from scratch to support the concepts introduced in the DRRT algorithm. This framework depends on the *RRT_BiasedLimCon* algorithm to generate the initial path and to re-grow the tree when fixing the path. This algorithm is modified further to support re-growing from an existing tree instead of completely replacing the old one. Thus, the re-planner can use the same function when fixing the path. Appendix F lists the C++ code for the implemented re-planning algorithm.

The framework is implemented as an event based command. Where for each simulation step the commend calls a PQP collision query for the next step. When a collision occurs, we mark all the parts of the tree that are invalidated by this collision. Then, the tree is trimmed to remove all these invalid parts. At this point, all the nodes and edges remaining in the tree are guaranteed to be valid, but the tree may no longer reach the goal. Finally, the tree is grown out until the goal is reached once more.

4.5.4 Visualization

As mentioned in Section 4.5, two main classes are implemented for taking care of visualizing the results on the crane directly in addition to its C-space. For visualizing the results on the crane, the *RawRecorder* class plots computed path as key frames on each

DoF of the crane. The 3D software is used then to run a simulation based on the applied key frames and render the scene in real-time.

Real-time rendering is done using the OpenGL renderer supported by the 3D software. Lighting and shading of the scene elements are done by developing OpengGL real-time shaders using the *Render Tree* in Softimage. For implementing standard lighting and shading models (e.g. Phong, Blinn, Lambert) a library of shaders is available. In case of requiring custom shaders that are not available in the shaders library, Softimage supports programmable vertex and pixel shaders for developing any custom shader.

4.6 VALIDATION AND EVALUATION

4.6.1 Description of the Case Studies

In this section the implemented algorithms are evaluated using a case of hydraulic crane with four DoFs. In addition, following the same procedure discussed on hydraulic cranes, another test is done for the case of a tower crane with 3 DoFs. Appendix B shows part of the procedure for calculating the tower crane homogenous transformation matrix based on DH-notation. The environment setting contains a steel frame structure that is composed of 536 members that are constructed as 3D polygonal meshes with 6360 triangles. These members are considered static as obstacles and are all parented under the *Obstacles* model.

For the dynamic obstacles, another working crane is added into the environment as a dynamic obstacle in the virtual construction site near the original crane so it is conflicting spatially with it. For the hydraulic crane case, each crane model consists of 20 objects and 15226 triangles. Thus, for the two cranes there will be 30452 triangles that need to be considered during collision detection calculations at each simulation step. For the case of tower cranes, each crane is composed of two objects and 86 triangles.

4.6.2 Validating the Results of the Case Studies

Figure 4.15 shows the environment in addition to the simulated hydraulic crane along with the initial and goal configurations. Crane with red material (left sided) is considered as a dynamic obstacle that can be pre-planned as a high priority crane or manipulated interactively while the simulation is running, while blue obstacles (steel frame structure) are static. In the captured snapshots, the simulated crane starts executing the path in order to move from its initial configuration to its goal configuration, both of which are shown in simulation step 1. At simulation step 120, the crane detects another crane as a dynamic obstacle and a re-planning phase is directly executed. The re-planning updates the original path of the crane to avoid the detected obstacle and continue executing the path until it reaches the goal at simulation step 240.

Figure 4.16 shows the same environment applied to the tower crane case study where the left sided crane is the dynamic obstacle for the right simulated one. At simulation step 1, the crane starts executing the path towards the goal shown at this step. As the crane advances towards the goal, it detects a collision with the other crane at simulation step

150. The re-planning phase is then executed and the path is updated directly. The new path drives the crane to go through the other direction in order to reach the goal as shown in simulation steps (150-400).

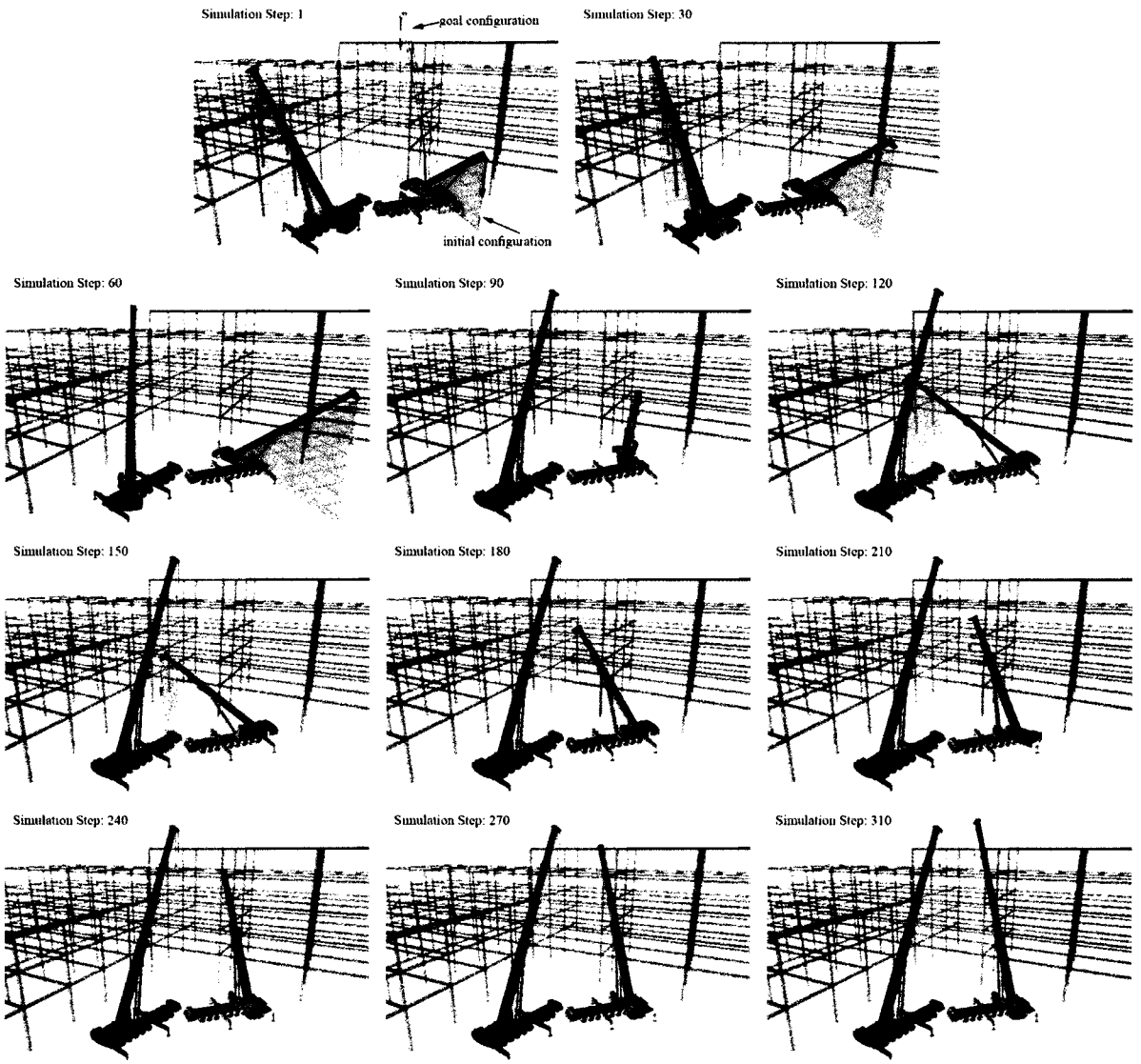


Figure 4.15: Simulation snapshots for the hydraulic crane case study

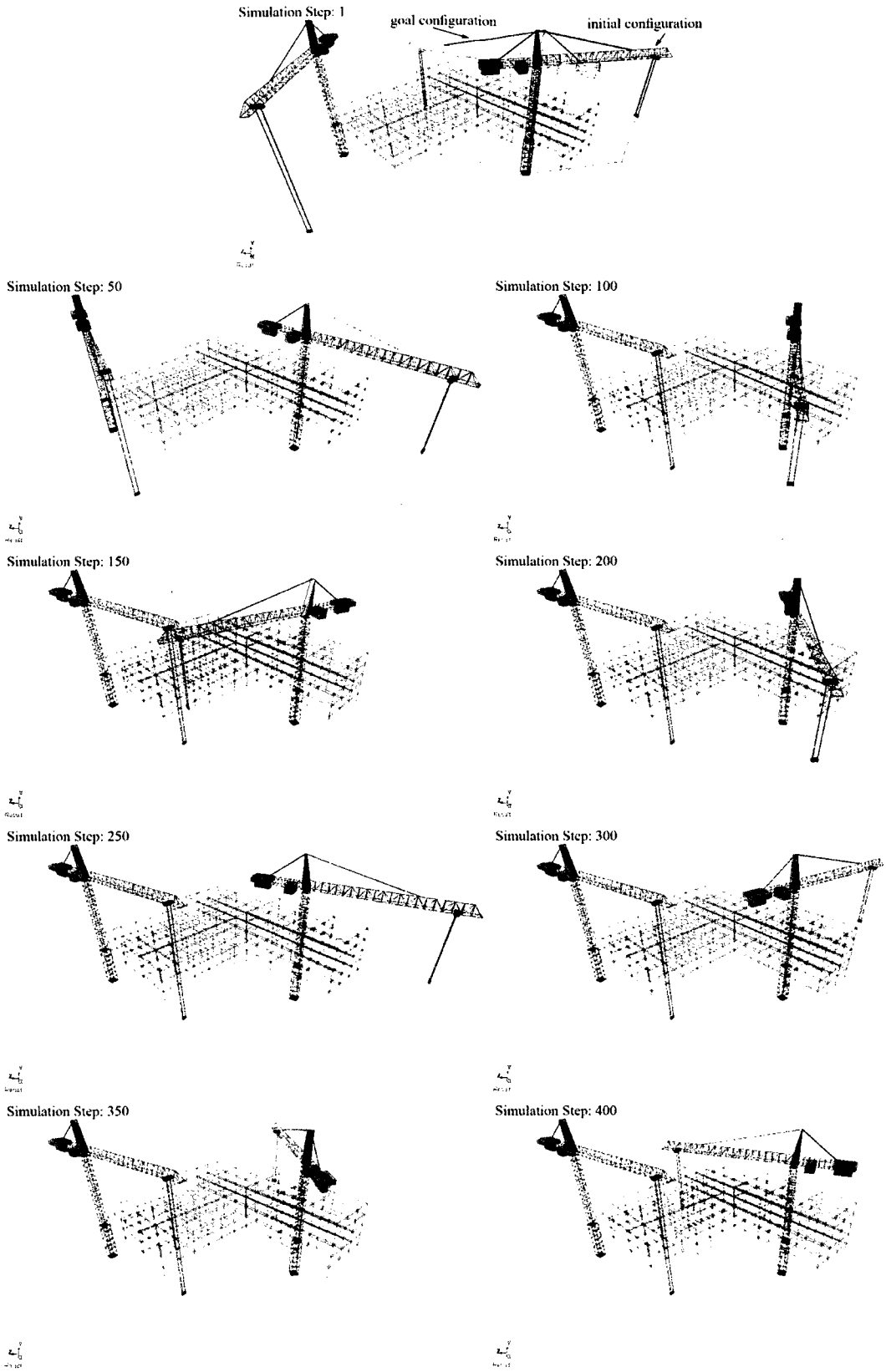


Figure 4.16: Simulation snapshots for the tower crane case study

4.6.3 Evaluating the Results of the Case Studies

The previous simulation tests were executed using the following parameter settings for the algorithms as shown in Table 4.1. Each test was repeated 50 times with different random seeds to evaluate the results with different random trees. Table 4.2 and Table 4.3 show the results summery for both planning and re-planning for the two crane cases.

Table 4.1: Values used in the evaluation tests

Parameter	Hydraulic Crane	Tower Crane
Max_Nodes	1000	500
Error_Tolerance	2	2
CSpace_Dilation (cm)	15	15
Space_Resolution ($hcube_{sub}$)	50	150
Biasing_Probability	0.1	0.1
Replan_Max_Nodes	2000	1000
Replan_Biasing_Probability	0.15	0.15

Table 4.2: Results summery for planning both case studies

	Hydraulic Crane	Tower Crane
Average for planning time (sec.)	5.256	1.071
Standard deviation for the planning time (sec.)	3.963	1.358
Average number of successful nodes	85.600	64.980
Standard deviation for the number of successful nodes	74.421	112.284
Average number of nodes on path	8.100	8.420
Standard deviation for the number of nodes on path	2.685	4.155

Table 4.3: Results summery for re-planning both case studies

	Hydraulic Crane	Tower Crane
Average for re-planning time (sec.)	1.736	0.035
Standard deviation for the re-planning time (sec.)	3.831	0.819
Average number of successful nodes	24.238	19.22
Standard deviation for the number of successful nodes	21.253	33.56
Average number of nodes on path	4.600	4.130
Standard deviation for the number of nodes on path	1.546	1.966

The reported times for the performance are based on the CPU time for an Intel T5550 Core2Duo processor 1.83 GHz. The planning time is the CPU time required to compute the initial path from the initial location of the crane to its goal. The re-planning time is the total CPU time for all re-planning operations needed to move the crane from the initial location to the goal.

In Table 4.2, the duration for calculating the initial path for the hydraulic crane was nearly 5 times the duration for the tower crane. Two main factors are responsible for the difference in planning time between the two cases; the first one is dimensionality, where in the case of the hydraulic crane the solver has to solve a four-DoF robot while for the tower crane it is only three. The other factor is the motion constraints evaluation that is applied in the hydraulic crane case and not applied in the tower crane case. These constraints include several mathematical expressions for simplifying the boom structure as explained in Section 4.4.1 in addition to the motion constraint for controlling the

direction of the cable which is discussed in Section 4.4.1. The number of successful nodes for the two cases is relatively near, this is because they are both solved in the same environment, the hydraulic crane uses more nodes to calculate the path because of the extra dimension it has. In both cases, the number of nodes on the path was in the minimum extents, and the results were all applicable paths that can be executed by a crane operator or by an automated robotic system that can control the crane. In Table 4.3, the re-planning time is the time taken for updating the path to avoid the detect obstacle. It shows acceptable results that can be improved by using powerful hardware. Since the re-planning phase is using the same algorithm, the number of nodes on the path to drive the equipment from its current location (location where the dynamic obstacle is detected) to the goal is efficient and applicable for cranes. Figure 4.17 shows 20 out the 50 paths that are generated for the hydraulic crane case. These paths are generated using different random seeds, and are visualized in the workspace. It is clear that even with different random seeds, all paths that are shown in the figure follow the same pattern which is the envelope for the optimal solution of the crane's case. This pattern can be divided into three main groups of action as shown in the figure. Action group 1 represents mainly the first part of the path that controls the boom extension. The average path length for this action group was 12.215m with a standard deviation of 5.069m. Action group 2 represents the major part of the path that controls the swing angle towards the goal. Its average path length was 31.916m with a standard deviation of 8.855m. The last part of the path represented in action group 3, looks less regular and has more nodes than the previous parts. This is because during this part of the path, the crane has to be controlled along different combinations of boom extension and changes of the angle to the ground to

avoid the other crane that is detected as a dynamic obstacle while executing the initial plan. The average path length for this action group was 6.265m with a standard deviation of 4.152m.

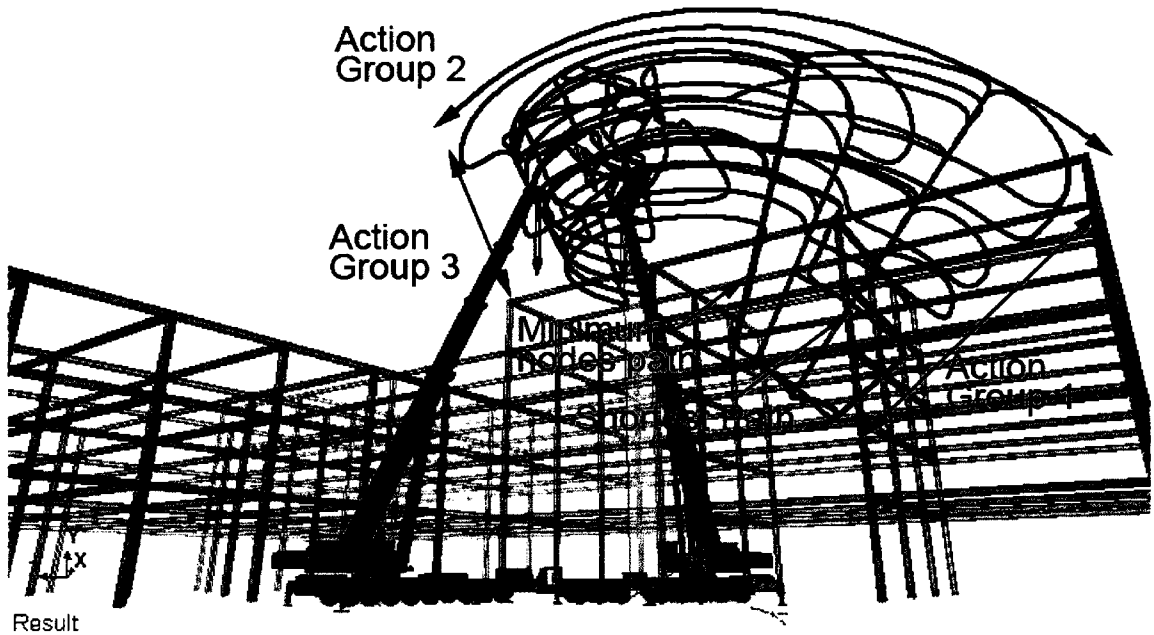


Figure 4.17: Different feasible paths generated for the hydraulic crane case study

For analyzing the actual time for each component of the system, the integrated Scene Debugger (Figure 4.18) is used to calculate the timing for each part of the computations. The scene debugger is an integrated tool in Softimage that allows analyzing the performance of scenes. This can tell the processing time and memory consumed by the elements in the scene, and help finding and correcting major performance bottlenecks.

Based on the Scene Debugger results, the percentage of average time consumed by the planning algorithm for the hydraulic case was 74.4% while the rest was consumed by

other tasks such as rendering and evaluating mathematical expressions and constraints. Based on the average duration for the total planning time, 3.9 seconds are consumed by the planning algorithm. This leads to understanding that the algorithm performance is applicable in real-time even with more complex cases and still having reasonable computation times. Appendix I shows the complete performance results from the Scene Debugger for one case.

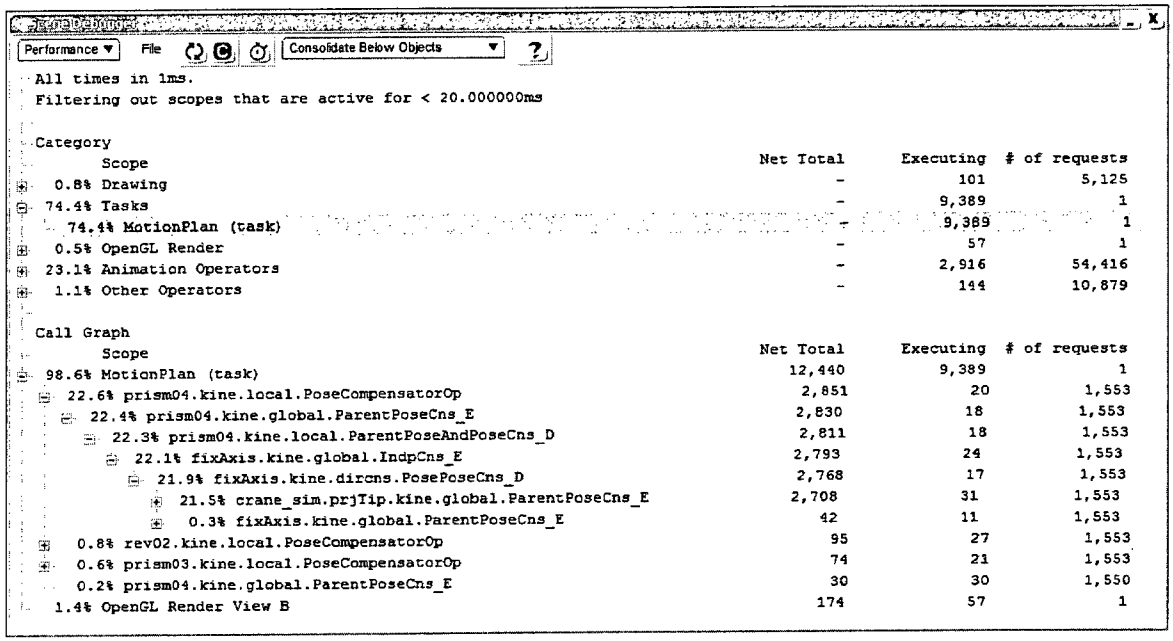


Figure 4.18: Screen capture for the Scene Debugger showing performance results

4.6.4 Effect of the Biasing Probability

For such randomly based algorithms, biasing the random sampling towards the goal has direct effect on the both the performance and the quality of the path. To analyze the relationship between the probability and both the performance and path quality: (1) the planning time; (2) the number of nodes in the tree; and (3) the number of nodes in the

path are plotted while the biasing probability is increased gradually from 0 to 1. To isolate other factors from affecting the tree behavior under different biasing probabilities, the tests were done using the proposed algorithm and the hydraulic crane case without obstacles. Figure 4.19 shows the testing results.

It is clearly noticed from the graph how the performance of the algorithm is enhancing as the value of the biasing probability is increasing. Between probability of 0.05 and 0.45 the performance enhanced with the factor of 5. However it is not recommended to have high value for the biasing probability because this limits the tree from sampling the C-space and makes it growing focused towards the goal. This behavior causes the tree to fall in a local minima in complex environments and fail to return a path as the case with the potential fields explained in Section 2.6.2.

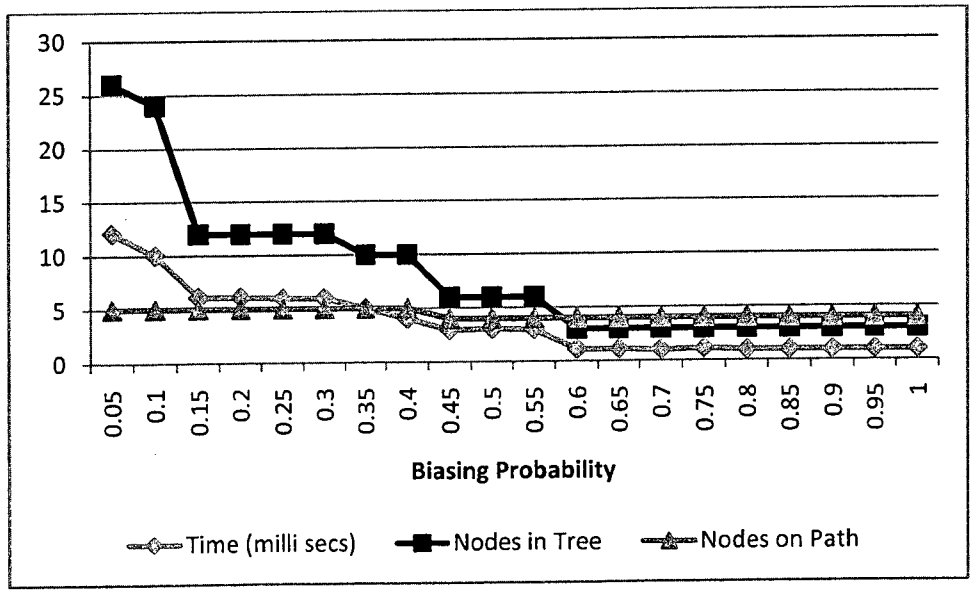


Figure 4.19: Relationship between the biasing probability and the proposed algorithm performance and results

4.6.5 Comparison with the Basic RRT Algorithm

For evaluating the performance of the enhanced algorithms, several comparison tests were done between the proposed enhanced algorithm *RRT Biased Limited Connect* and the standard RRT with goal biasing that is used in the original DRRT algorithm. These tests were done using the hydraulic crane case study with the same environment setting. Figure 4.20 shows the planning time for 50 tests with different random seeds. The average performance of the *RRTGoalBiased* was 5.1167 seconds while for the *RRTBiasedLimCon* was 1.0733 seconds. These results show a significant enhancement in the performance of the *RRTBiasedLimCon* over the *RRTGoalBiased* with the factor of 5. Figure 4.21 and Figure 4.22 shows the results for the same 50 tests for the number of nodes in the tree and on the path. The average number of nodes in of the *RRTGoalBiased* was 1590.5 and it was 43 times the number of nodes in the *RRTBiasedLimCon* where its average was 36.88. Since the proposed algorithm depends on a limited greedy connect heuristic, the number of nodes on the path was minimal in comparison with standard RRT algorithm. The average number of nodes on the path for the *RRTBiasedLimCon* 8.48 was while for the *RRTGoalBiased* was 187.48 which is around 22 times of the proposed algorithm.

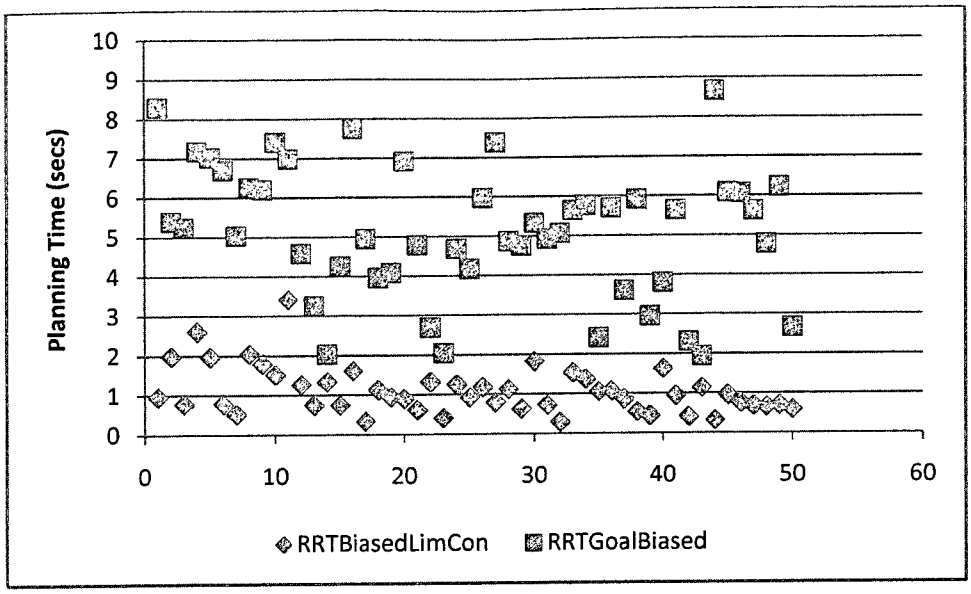


Figure 4.20: Planning time of the two compared algorithms

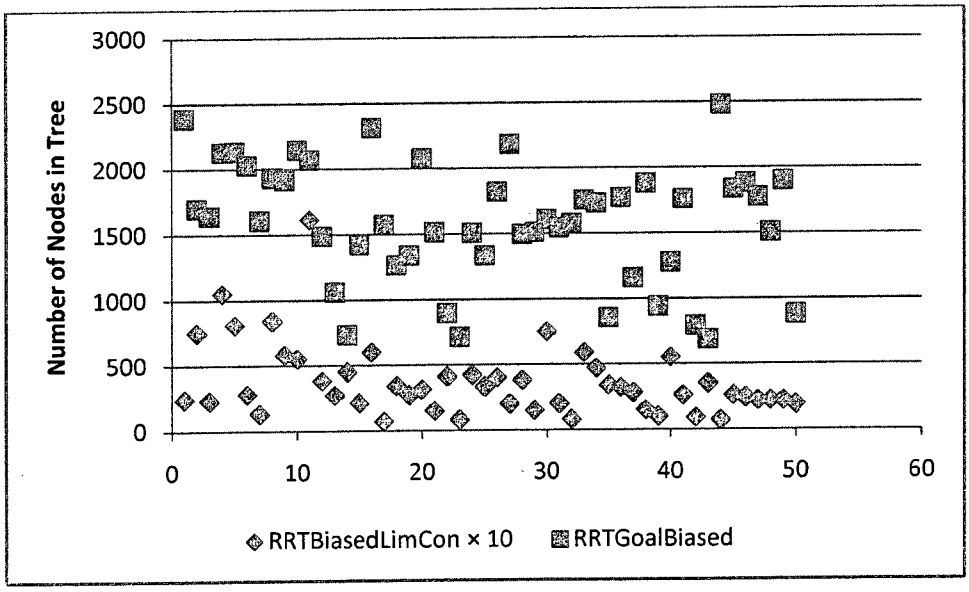


Figure 4.21: Number of nodes in the tree of the two compared algorithms

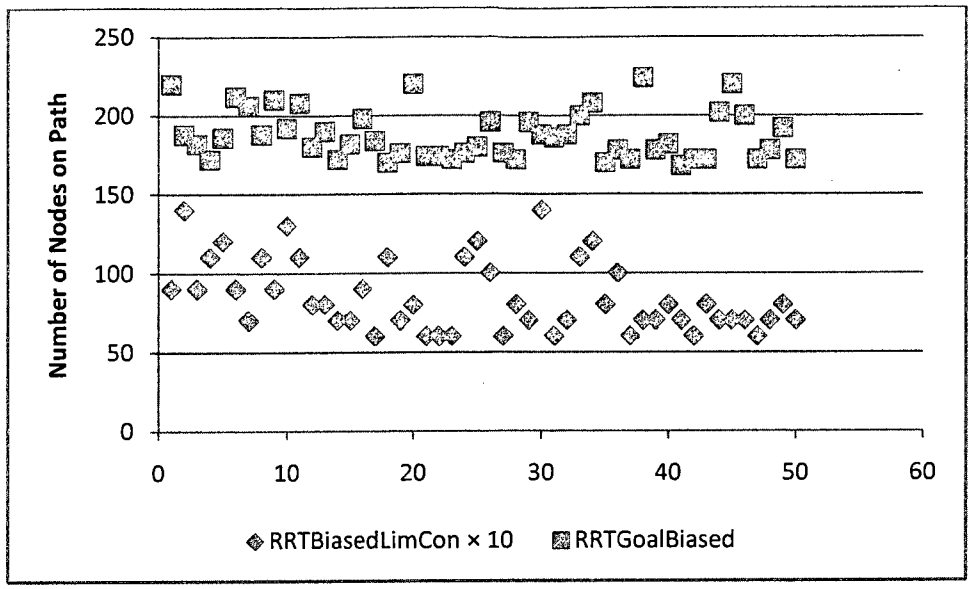


Figure 4.22: Number of nodes on the path of the two compared algorithms

4.7 SUMMARY AND CONCLUSIONS

This chapter described the implementation of the proposed approaches and methods discussed in Chapters 3. A prototype system is developed and two case studies of a hydraulic crane and a tower crane motion planning were used to demonstrate the feasibility of these approaches and methods. The software development tools were selected to integrate several information technologies in the prototype system. The prototype was integrated into a 3D software package to provide high modeling and kinematic structuring capabilities that are derived from the 3D software. Several optimization techniques are proposed to enhance the efficiency of the prototype and enable real-time visualizing and simulation. The developed prototype system was

evaluated and validated intensively on both case studies. The results show good indicators for the applicability of the system in near real-time motion planning for construction equipment by integration with suitable operating hardware and tracking sensors that are able to return accurate information in near real-time. The biasing probability factor was analyzed to show its effect on the overall performance of the algorithm. Several comparison tests were made between the proposed *RRTBiasedLimCon* algorithm and the basic RRT algorithm with goal biasing. The results show that the proposed algorithm reduces the planning time with an average factor of 5, the number of nodes in the tree with an average factor 43 and the number of nodes on the path with an average factor of 22.

In summary, the real-time motion planning system is useful in helping crane engineers and project managers to plan and validate equipment motion in complex environments and finding spatial conflicts for specific tasks within a certain period. For equipment operators, the same system can provide intelligent decision support for re-planning in real-time while the equipment is executing its plan. Environment updates are assumed to be retrieved from the environment using any suitable real-time capturing hardware.

CHAPTER 5 SUMMARY, CONCLUSIONS AND FUTURE WORK

5.1 SUMMARY

This thesis investigated several algorithms and methods from artificial intelligence, robotics and computer graphics literature to developed a new framework that can plan, simulate and visualize cranes and other construction equipments in real-time. Construction equipments were treated as robots consisting of moving rigid parts and mathematical expressions were derived for modeling the kinematic structure of these equipments. The feasibility of the proposed framework was illustrated with two examples of a hydraulic crane, which is treated as a four-DoFs robot, and a tower crane with three DoFs. Using the same principles and procedures, kinematical models for other types of heavy construction equipments can be derived.

Because motion planning algorithms are mainly developed for robotics applications, they do not consider engineering constraints and the collision-free paths they find may not be easy to apply in case of construction equipment. A new motion planning solver has been developed in this research that satisfies construction equipment constraints. The core part of this solver is a computer algorithm, *RRT_BiasedLimCon*, which was developed and implemented to find collision-free paths that can be followed by cranes to transport construction elements both safely and efficiently. An engineering agent approach is developed to ensure that the collision-free paths are realistic and safe in terms of engineering constraints.

This research also developed and implemented an efficient algorithm for real-time motion planning that is able to consider the dynamic features found in construction environments and other equipments in case of multi-equipment planning. In all cases, motion planning is significantly more complex than the case of static known environments because of two main reasons: (1) based on the information of new obstacles detected during executing the paths, the motion planning algorithm must be able to generate a modified plan in real-time to cope with the new detected obstacles; and (2) the new obstacles being detected could be moving in the site and their movement is unknown and unpredictable. The tree data structure generated by the developed algorithm to capture and avoid the obstacles is modified to allow minimum amount of computation when new obstacles are detected.

A prototype system, Intelligent Construction Equipment motion Planner (ICE-Planner), was developed to implement all previously described computational methods and functions to plan and simulate heavy construction equipments (mainly cranes). The system was integrated into a 3D software to provide a flexible environment for modeling, interacting with and visualizing the full simulation. The system includes several Graphical User Interfaces (GUIs) to set any parameter related to the system in addition to visualizing the C-space generated while computing the path. Two case studies were created to demonstrate the applicability of the proposed framework and algorithms using models of a tower crane and a hydraulic crane. They were also used to validate and evaluate the performance of each component of the system. The hydraulic crane model of TMS870 Grove Crane was prepared along with a database for part of its load charts. The

cranes were simulated in a steel frame construction environment in addition to a second crane working nearby as a dynamic obstacle. The results of the preliminary testing of the system showed the ability of generating realistic plans in addition to rapid re-planning.

5.2 CONCLUSIONS

The conclusions of this research are grouped into the following areas:

(1) A motion planning framework for supporting construction equipment to ensure safety and improve productivity has been proposed. Although this framework was applied and tested using hydraulic and tower cranes, it is designed to have the flexibility to work for other types of heavy construction equipments by avoiding engineering of the system towards the functions of specific equipment type.

(2) Several methods have been introduced to ensure that the generated paths satisfy safety requirements when applied to heavy construction equipment. These methods are the integration of engineering constraints and rules of actions into the motion planning process, and the introduction of critical volumes and obstacles dilation for avoiding unsafe paths.

(3) An efficient planning/re-planning algorithm has been developed for dynamic environments which is able to provide information for operators to assist them in manipulating equipments more efficiently and more safely. Additionally, this algorithm is applied to consider multiple equipments efficiently by doing prioritized motion planning for each equipment. This scenario is tested in the case

studies by taking the high priority crane into consideration in the simulation as a dynamic obstacle for the low priority crane.

(4) A prototype system was implemented to demonstrate the feasibility of the proposed framework and algorithms using two case studies. Several optimization techniques are proposed to enhance the efficiency of the prototype and enable real-time visualizing and simulation. The prototype was integrated into a 3D software package to provide high modeling and kinematic structuring capabilities that are derived from the 3D software. Intensive evaluation and validation tests were done to demonstrate the applicability of system in real-time motion planning for construction equipment. The biasing probability factor was analyzed to show its effect on the overall performance of the algorithm. Several comparison tests were made to show the enhancements in efficiency and paths results between the proposed *RRTBiasedLimCon* algorithm and the basic RRT.

5.3 LIMITATIONS AND FUTURE WORK

While fulfilling the objectives of this research, there are many areas that require further research including the following:

(1) Since the planning algorithm is based on randomized sampling of the C-space, the generated path can be enhanced further by doing post processing to optimize it and remove redundant nodes. The optimized paths should be validated against engineering constraints to guarantee realistic paths. The results of the optimization process has a minor effect on the path optimality since the original

generated path is near optimal, thus results of this optimization step should be also evaluated in terms of performance in the case of replanning where rapid response has a higher priority than minor path enhancements.

(2) Although methods and tools developed in this research are engineered to solve general cases of heavy construction equipments. Currently ICE-Planner is validated only for hydraulic and tower cranes. In actual construction projects, other types of equipments can be used and it is important to consider them in the system. Further testing and evaluation with other types of construction equipments is required to prove this feature of the system.

(3) Although the proposed framework supports physical based motions (e.g. swing motion of the cable) applied to any part of the equipment by implementing differential equations on the kinematic structure, it is not included in the case study presented in this research. Future research should be done to consider different types of physical based motions/deformations in cranes (e.g. boom deflection of hydraulic cranes when lifting heavy weights).

(4) Although the proposed framework is able to consider multiple equipments working separately as dynamic obstacles (prioritized multi-equipment planning), it is not designed to directly support collaborative multiple equipments, such as the case of two cranes working together on the same task. This issue requires more research in the following areas: (a) Creating a kinematic model for the collaborative equipment case to be solved by the proposed system as a centralized motion planning, where the solver will treat the collaborative equipments as one chained, closed robot that has the total number of DoFs equals to the sum of the

DoFs of both equipments; (b) Developing methods and techniques for simplifying the high complexity that arises from solving centralized motion planning; and (c) Evaluating the efficiency of the modified algorithms for solving such complex cases.

(5) The proposed framework can be extended to provide the basis for developing solutions to other areas related to construction equipments such as the optimization of the location of the heavy construction equipment. This can be approached with the work introduced here by using the parameterization results of the generated paths as the fitness function in a Genetic Algorithm (GA). GA will then evolve a solution for the optimized location of the equipment by evaluating the paths generated by the motion planning algorithm.

(6) Building Information Model (BIM) can be used to automatically update the construction environment through time and generate incremental path planning queries for the whole project phases. This can be done by allocating each crane to its group of construction elements that it will handle sequentially. Further research is required in this area for arranging the BIM data and importing it into the proposed system. Additionally, multi-task planning using BIM can be considered to enhance the performance of sequential motion planning queries by caching previous generated RRTs and updating them based on site updates.

(7) The DRRT requirement of growing the tree reversed makes it hard to utilize other efficient enhancements developed for the RRT such as simultaneously growing two trees rooted at the initial and the goal configurations (forward-tree and backwards-tree). Thus, in cases where new obstacles are discovered while

traversing through the forward tree (the tree that grows from the initial configuration), a large part of the entire tree needs to be re-grown. This issue requires more research to analyze and compare the performance between having single tree DRRT and dual-tree brute-force RRT.

REFERNCES

Abrahams, D. et al. (2003). Technical Report on C++ Performance, The Standard Committee's Technical Report on Implementation Issues and Programming Techniques Related to Performance.

Ahuactzin, J.M., Talbi, E.G., Bessi`ere, P., and Mazer, E. (1992). Using Genetic Algorithms for Robot Motion Planning. In proceedings of the 10th European Conference of Artificial Intelligence, London, England, pp. 671-675.

Akenine-Moller, T., Haines, E. and Hoffman, N. (2008). Real-Time Rendering Third Edition. A K Peters, Ltd., Wellesley, MA.

Al-Hussein, M. (1999). An Integrated Information System for Crane Selection and Utilisation, Ph.D. thesis, Concordia University.

Al-Hussein, M., Alkass, S., and Moselhi, O. (2000). D-CRANE: A Database System for Utilization of Cranes, Canadian Journal of Civil Engineering, Volume 27, No.6, pp. 1130-1138.

Al-Hussein, M., Alkass, S., and Moselhi, O. (2005). Optimization Algorithm for Selecting and on Site Location of Mobile Cranes, Journal of Construction Engineering and Management (ASCE), Volume 131, No. 5, pp. 579-590.

Ali, M.S.A.D., Babu, N.R. and Varghese, K. (2005). Collision Free Path Planning of Cooperative Crane Manipulators using Genetic Algorithm. Journal of Computing in Civil Engineering, ASCE, Volume 19, No, 2, pp. 182-193.

Amenta, N., Asano, T., Barequet, G., Bern, M., Boissonat, J.D., Canny, J., Chazelle, B., Clarkson, K., Dobkin, D., Donald, B., Drysdale, S., Edelsbrunner, H., Eppstein, D., Forrest, R., Fortune, S., Goldberg, K., Goodrich, M., Guibas, L., Hanrahan, P., Hoffman, C., Huttenlocher, D., Imai, H., Kirkpatrick. D., Lee, D.T., Mehlhorn, K., Milenkovic, V., Mitchell, J., Overmars, M., Pollack, R., Seidel, R., Sharir, M., Snoeyink, J., Toussaint, G., Teller, S., Voelcker, H., Welzl, E., and Yap, C. (1996). Application Challenges to Computational Geometry, Advances in Discrete and Computational Geometry – Proceedings of AMS-IMS-SIAM Joint Summer Research Conference Discrete and Computational Geometry: Ten Years Later, Contemporary Mathematics, No. 223, pp. 407-423.

Bahnassi, H. (2009). Scripting Structures. <http://www.xsi-blog.com/archives/116>, Accessed in Dec, 2009.

Ban, Y. (2002). Unmanned Construction System: Present Status and Challenges. Proceedings of the 19th International Symposium on Automation and Robotics in Construction, Washington, U.S., pp. 241-246.

Barraquand, J. and Latombe, J.C. (1991). Robot Motion Planning: A Distributed Representation Approach, International Journal of Robotics Research, Volume 10, No. 6, pp. 628-649.

Barraquand, J., Kavraki L., Latombe, J.C., Li, T.Y., Motwani, R., and Raghavan, P. (1997). A Random Sampling Scheme for Path Planning, International Journal of Robotics Research, Volume 16, Issue 6, pp. 759-774.

Bennewitz, M., Burgard, W., Thrun S. (2002). Finding and Optimizing Solvable Priority Schemes for Decoupled Path Planning Techniques for Teams of Mobile Robots. Robotics and Autonomous Systems, Volume 41, No.2, pp. 89-99.

Bien, Z. and Lee, J. (1992). A Minimum-Time Trajectory Planning Method for Two Robots, IEEE Transactions on Robotics and Automation, Volume 8, No.3, pp. 414-418.

Bouvel, D., Froumentin, M. and Garcia, G. (2001). A Real-Time Localization System for Compactors, Automation in Construction, Vol. 10, pp. 417-428.

Brandt, D. (2006). Comparison of A* and RRT-Connect Motion Planning Techniques for Self-Reconfiguration Planning, Proceedings of the 2006 IEEE/RSJ International Conference on Intelligent Robots and Systems, October, China.

British Standard (2000), Code of practice for safe use of cranes - Mobile cranes, BS 7121-13.

Bruce, J. (2009). Robot Motion Planning Slides from Computer Science Department Carnegie Mellon University, <http://www4.cs.umanitoba.ca/~jacky/Robotics/Papers/Bruce-ERRTMotionPlanningSlides.pdf >, Accessed in Dec, 2009.

Bruce, J. and Veloso, M. (2002). Real-time Randomized Path Planning for Robot Navigation, in Proceedings of the IEEE International Conference on Intelligent Robots and Systems (IROS).

Canny, J. (1988). The Complexity of Robot Motion Planning. MIT Press, Cambridge, MA.

Cheng, P. and LaValle, S.M. (2002). Resolution complete rapidly-exploring random trees, Proceedings of International Conference on Robotics & Automation, Washington, D.C., U.S. Volume 1, pp. 267-272.

Chi, H.L., Hung, W.H. and Kang, S.C. (2007). A Physics Based Simulation for Crane Manipulation and Cooperation, Computing in Civil Engineering 2007, Proceedings of the 2007 ASCE International Workshop on Computing in Civil Engineering, pp. 777-784.

Choset, H., Lynch, K.M., Hutchinson, S., Kantor G., Burgard, W., Kavraki, L.E. and Thrun, S. (2005). Principles of Robot Motion – Theory, Algorithms, and Implementations, the MIT Press, Cambridge.

Clark, C.M., Bretl, T. and Rock S. (2002). Applying Kinodynamic Randomized Motion Planning with a Dynamic Priority System to Multi-Robot Space Systems, Proceedings of IEEE Aerospace Conference, pp. 3621-3631.

Craig, J. (2004). Introduction to Robotics Mechanics and Control Third Edition. Prentice Hall, New Jersey.

Crane Accident Statistics 2009. <http://www.craneaccidents.com/stats.htm>, Accessed in Dec, 2009.

Cranimation (2009). <http://www.cranimax.com>, Accessed in Dec, 2009.

CSST (2009). Commission de la santé et de la sécurité du travail du Québec, <http://www.csst.qc.ca/portail/fr>, Accessed in Dec, 2009.

Denavit, J. and Hartenberg, R.S. (1955). A Kinematic Notation for Lower-Pair Mechanism Based On Matrices, Journal of Applied Mechanics; Volume 77, pp. 215-221.

DirectX (2009). <http://msdn.microsoft.com/en-us/directx/default.aspx>, Accessed in Dec, 2009.

Dudek, G. and Jenkin, M. (2000). Computational Principles of Mobile Robotics, Cambridge University Press, New York.

Erdmann, M. and Lozano-Perez, T. (1987). On Multiple Moving Objects. Algorithmica, Volume 2, No.4, pp. 477-521.

Ericson, C. (2005). Real-Time Collision Detection, Morgan Kaufmann, San Francisco.

Fair, H.W. (1998). Crane Safety on Construction Sites, ASCE manuals and reports on engineering practice, No. 93.

Ferguson, D., Kalra, N., and Stentz, A. (2006). Replanning with RRTs. Proceeding of the IEEE International Conference on Robotics and Automation, pp. 1243-1248.

Fried, J., Davydov, E., and Pa, W. (2009). Robotics and Motion Planning, A Sophomore College Presentation, Stanford Computer Science Education. <http://cse.stanford.edu/class/sophomore-college/projects-98/robotics/>, Accessed in Dec, 2009.

Furlani, K.M., Latimer, IV D.T., Gilsinn, D.E. and Lytle, A.M. (2002). Prototype Implementation of an Automated Structural Steel Tracking System, Proceedings of the 19th International Symposium on Automation and Robotics in Construction, Maryland, September, pp. 467-473.

Groove Crane. (2008). TMS870/TTS870 Product Guide, Manitowoc Crane Group.

Hart, P.E., Nilsson, N.J. and Raphael, B. (1968). A Formal Basis for the Heuristic Determination of Minimum Cost Paths, IEEE Transactions on Systems Science and Cybernetics, Volume 4, Issue 2, pp. 100-107.

Hornaday, W.C., Haas, C.T., O'Connor, J.T., Wen, J. (1993). Computer-Aided Planning for Heavy Lifts, Journal of Construction Engineering and Management, Volume 119, No. 3, pp. 498-515.

Integrated Publishing (2009). <http://www.tpub.com/content/engine/14081/css/14081_331.htm>, Accessed in Dec, 2009.

Kamat, V.R. and Martinez, J.C. (2001). Visual Simulated Construction Operations in 3D. Journal of Computing in Civil Engineering, ASCE, Volume 15, No. 4, pp. 329-337.

Kang, S.C. (2006). A Generic Method to Model Construction Cranes, EASEC-10, Bangkok, Thailand, August 3-5.

Kang, S.C. and Miranda, E. (2006). Planning and Visualization for Automated Robotic Crane Erection Processes in Construction, Automation in Construction, Volume 15, Issue 4, pp. 398-414.

Kavraki, L., Latombe, J.C. (1998). Probabilistic Roadmaps for Robot Path Planning, Practical Motion Planning in Robotics: Current Approaches and Future Directions, pp. 33-53.

Kavraki, L., Latombe, J.C., Motwani, R., and Raghavan, P. (1995). Randomized Query Processing in Robot Motion Planning. In Proceedings ACM symposium on Theory of Computing, pp. 353-362.

Kim, S.K., Russell, J.S. and Koo, K.J. (2003). Construction Robot Path-Planning for Earthwork Operations, Journal of Computing in Civil Engineering, Volume 17, No. 2, pp. 97-104.

Koditschek, D. (1987). Exact Robot Navigation by Means of Potential Functions: Some Topological Considerations, Proceedings. 1987 IEEE International Conference on Robotics and Automation, pp. 1-6.

Kondo, K. (1991). Motion planning with Six Degrees of Freedom by Multistrategic Bidirectional Heuristic Free-Space Enumeration, IEEE Tr. on Robotics and Automation, Volume 7, No. 3, pp. 267-277.

Kuffner, J.J. and LaValle, S.M. (2000). RRT-Connect: An Efficient Approach to Single-Query Path Planning, in Proceedings of the IEEE International Conference on Robotics and Automation (ICRA).

Kumi A.T. and Retik A. (1997). Library-based 4D Visualization of Construction Processes, Proceedings of the IEEE Conference on Information Visualisation, Piscataway, NJ, pp. 315-321.

Laia, K.C. and Kang, S.C. (2009). Collision Detection Strategies for Virtual Construction Simulation. Automation in Construction Volume 18, Issue 6, October 2009, pp. 724-736.

Latombe, J.C. (1991). Robot Motion Planning. Kluwer, Boston, MA.

Latombe, J.C. (2009). Motion Planning Slides from Stanford University CS326A Course <http://robotics.stanford.edu/~latombe/cs326/2009/class3/class3.htm>, Accessed in Dec, 2009.

LaValle, S.M. (1998). Rapidly-Exploring Random Trees: A New Tool for Path Planning. TR 98-11, Computer Science Dept., Iowa State University.

LaValle, S.M. (2006). Planning Algorithms. Cambridge University Press, New York.

LaValle, S.M. and Kuffner, J.J. (1999). Randomized Kinodynamic Planning. Proceedings of Robotics and Automation, Volume 1, pp. 473-479.

LaValle, S.M., Hutchinson S.A. (1998). Optimal Motion Planning for Multiple Robots Having Independent Goals. IEEE Transactions on Robotics and Automation, Volume 14, No.6, pp. 912-925.

LiftPlanner (2009). <http://www.liftplanner.net>, Accessed in Dec, 2009.

Lorensen, W.E. and Cline, H.E. (1987). Marching Cubes: A High Resolution 3D Surface Construction Algorithm, Computer Graphics, Volume 21, No. 4, pp. 163-169.

Lozano-Perez, T. and Wesley, M. (1979) An Algorithm for Planning Collision-free Paths Among Polyhedral Obstacles. Communications of the ACM, Volume 22, Issue 10, pp. 560-570.

Lytle, A.M., Saidi, K.S. and Stone, W.C. (2002). Development of a Robotic Structural Steel Placement System, Proceedings of the 19th International Symposium on Automation and Robotics in Construction, pp. 263-268.

Lytle, A.M., Saidi, K.S., Bostelman, R.V., Stone, W.C. and Scott, N.A. (2004). Adapting a Teleoperated Device for Autonomous Control Using Three-Dimensional Positioning Sensors: Experiences with the NIST RoboCrane, Automation in Construction, Volume 13, pp. 101-118.

Manrique, J. D., Al-Hussein, M., Telyas, A. and Funston, G. (2007), Constructing a Complex Precast Tilt-Up-Panel Structure Utilizing an Optimization Model, 3D CAD, and Animation, Journal of Construction Engineering and Management, Volume 133, No. 3, pp. 199-207.

Motion Strategy Library (MSL) (2009). <http://msl.cs.uiuc.edu/msl/>, Accessed in Dec, 2009.

NIOSH (2009). The National Institute for Occupational Safety and Health, <http://www.cdc.gov/NIOSH/>, Accessed in Dec, 2009.

NIST (2009). <http://www.nist.gov>, Accessed in Dec, 2009.

O'Dunlaing, C. and Yap, C. (1982). A Retraction Method for Planning the Motion of a Disc, Journal of Algorithms, Volume 6, pp. 104-111.

O'Donnell, P.A. and Lozano-Perez, T. (1989). Deadlock-Free and Collision-Free Coordination of Two Robot Manipulators. In Proceedings of the IEEE International Conference on Robotics & Automation (ICRA).

Oloufa, A.A., Ikeda, M., and Oda, H. (2003). Situational Awareness of Equipment using GPS, Wireless, and Web Technologies. Automation in Construction, Volume 12, Issue 6, pp. 737-748.

OpenGl (2009). <http://www.opengl.org>, Accessed in Dec, 2009.

Peng, J., Akella, S. (2003). Coordinating the Motions of Multiple Robots with Kinodynamic Constraints. Proceedings of the IEEE International Conference on Robotics and Automation, pp. 4066-4073.

Peyret, F., Jurasz, J., Carrel, A., Zekri, E. and Gorham, B. (2000). The Computer Integrated Road Construction Project, Automation in Construction, Volume 9, pp. 447-461.

Proximity Query Package (PQP) (2009). <http://www.cs.unc.edu/~geom/SSV/>, Accessed in Dec, 2009.

Reif, J. (1979) Complexity of the Mover's Problem and Generalizations. In Proceeding of 20th IEEE Symposium on Foundations of Computer Science, pp. 421-427.

Rimon, E. and Koditschek, D. (1992). Exact Robot Navigation Using Artificial Potential Functions, IEEE Transactions on Robotics and Automation, Volume 8, Issue 6, pp. 501-518.

S´anchez, G., Latombe, J.C. (2002). Using a PRM Planner to Compare Centralized and Decoupled Planning for Multi-Robot Systems. Proceedings of IEEE International Conference on Robotics and Automation, pp. 2112-2119.

Saidi, K. and Lytle, A. (2008). NIST Research in Crane Automation: 2007 Overview, the 87th Annual Meeting, Transportation Research Board.

Schwartz, J.T., Sharir, M. (1983). On the Piano Movers' Problem III: Coordinating the Motion of Several Independent Bodies: The Special Case of Circular Bodies Moving Amidst Polygonal Obstacles, International Journal of Robotics Research, Volume 2, No. 3, pp. 46-75.

Sim´eon, T., Leroy, S. and Laumond J.P. (2002). Path Coordination for Multiple Mobile Robots: A Resolution Complete Algorithm. IEEE Transactions on Robotics and Automation, Volume 18, No.1, pp. 42-49.

Simlog (2009). <http://www.simlog.ca>, Accessed in Dec, 2009.

Sivakumar, P.L., Varghese, K. and Babu, N.R. (2003). Automated Path Planning of Cooperative Crane Lifts Using Heuristic Search. Journal of Computing in Civil Engineering, ASCE, Volume 17, No. 3, pp. 197-207.

Softamage (2009). <http://www.softimage.com>, Accessed in Dec, 2009.

Spong, M.W., Lewis, F.L. and Abdallah, C.T. (1992). Robot Control – Dynamics, Motion Planning, and Analysis, IEEE Press, IEEE Control Systems Society.

Stentz, A. (1994). Optimal and efficient path planning for partially-known environments, In Proceedings of the IEEE International Conference on Robotics and Automation, pp. 3310-3317.

Stentz, A. (1995). The Focussed D* Algorithm for Real-Time Replanning, In Proceedings of the International Joint Conference on Artificial Intelligence, Montreal, Quebec, pp. 1652-1659.

Stone, W.C. and Pfeffer, L.E. (1998). Automation Infrastructure System for a Robotic 30-ton Bridge Crane, Proceedings of the Third ASCE Specialty Conference on Robotics for Challenging Environments held in Albuquerque, New Mexico, April 26-30, pp. 195-201.

Stroustrup, B. (1997). The C++ Programming Language Third Edition. AddisonWesley, Reading, Massachusetts.

Tseng, Y.H., Chang, J.R., Kang, S.C. Tseng, C.H. and Wu, P.H. (2007). The Study in Using an Autonomous Robot for Pavement Inspection, International Symposium on Automation and Robotics in Construction, Kochi, Kerala, India. September 19-21.

Tserng, H.P., Ran, B. and Russell, J.S. (2000). Interactive Path Planning for Multi-Equipment Landfill Operations, Automation in Construction 10, pp. 155-168.

WorkSafeBC (2009). <http://worksafebc.com>, Accessed in Dec, 2009.

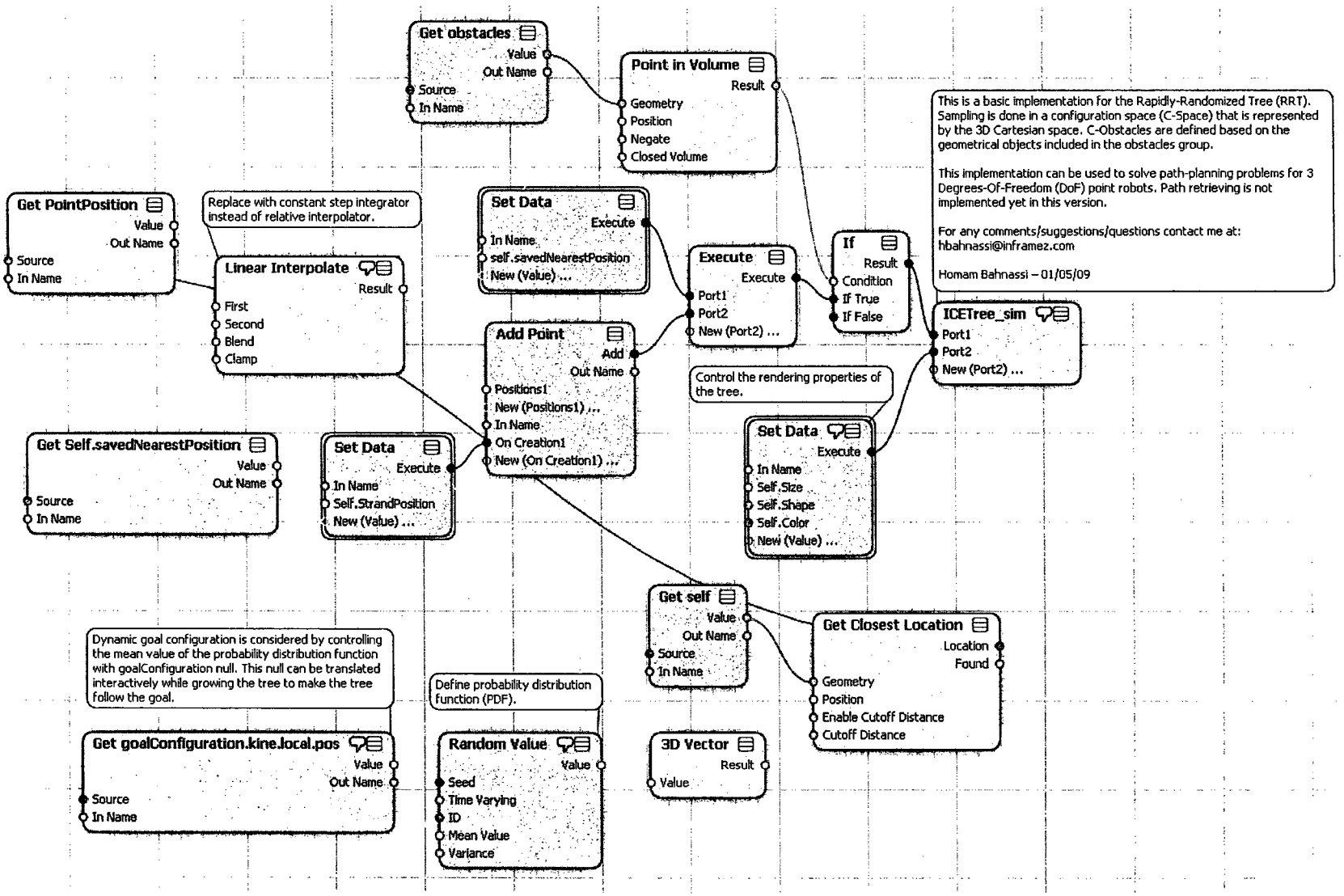
Zhang, C., Hammad, A., Bahnassi, H. (2009). Collaborative Multi-Agent Systems for Construction Equipment Based on Real-Time Field Data Capturing, Journal of Information Technology in Construction (ITcon), Volume 14, Special Issue Next Generation Construction IT: Technology Foresight, Future Studies, Roadmapping, and Scenario Planning, pp. 204-228.

Zhang, C., Hammad, A., Zayed, T.M., Wainer, G. and Pang, H. (2007). Cell-based Representation and Analysis of Spatial Resources in Construction Simulation, Journal of Automation in Construction, Volume 16, No. 4, pp. 436-448.

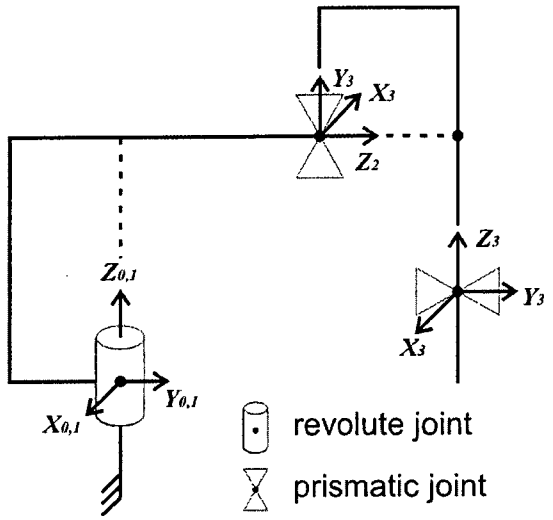
Zhou, S. and Zhang, S. (2007). Co-simulation on Automatic Pouring of Truck-mounted Concrete Boom Pump, Proceedings of the IEEE International Conference on Automation and Logistics, August 18-21, 2007, Jinan, China.

APPENDICES

APPENDIX A: Programming-tree for the RRT visualization prototype created using the visual programming system in Softimage: The prototype was created in this system by building the algorithm using abstract programming nodes that are connected together based on the logic of the implemented algorithm.


Get obstacles
Value
Out Name
Source
In Name
Point in Volume
Result
Geometry
Position
Negate
Closed Volume
This is a basic implementation for the Rapidly-Randomized Tree (RRT). Sampling is done in a configuration space (C-Space) that is represented by the 3D Cartesian space. C-Obstacles are defined based on the geometrical objects included in the obstacles group.
This implementation can be used to solve path-planning problems for 3 Degrees-Of-Freedom (DoF) point robots. Path retrieving is not implemented yet in this version.
For any comments/suggestions/questions contact me at: hbahnassi@inframez.com
Homam Bahnassi – 01/05/09
Get PointPosition
Value
Out Name
Source
In Name
Replace with constant step integrator instead of relative interpolator.
Linear Interpolate
Result
First
Second
Blend
Clamp
Set Data
Execute
In Name
self.savedNearestPosition
New (Value) ...
Execute
Execute
Port1
Port2
New (Port2) ...
If
Result
Condition
If True
If False
ICETree_sim
Port1
Port2
New (Port2) ...
Add Point
Add
Out Name
Positions1
New (Positions1) ...
In Name
On Creation1
New (On Creation1) ...
Control the rendering properties of the tree.
Set Data
Execute
In Name
Self.Size
Self.Shape
Self.Color
New (Value) ...
Get Self.savedNearestPosition
Value
Out Name
Source
In Name
Set Data
Execute
In Name
Self.StrandPosition
New (Value) ...
Get self
Value
Out Name
Source
In Name
Get Closest Location
Location
Found
Geometry
Position
Enable Cutoff Distance
Cutoff Distance
Dynamic goal configuration is considered by controlling the mean value of the probability distribution function with goalConfiguration null. This null can be translated interactively while growing the tree to make the tree follow the goal.
Define probability distribution function (PDF).
Get goalConfiguration.kine.local.pos
Value
Out Name
Source
In Name
Random Value
Value
Seed
Time Varying
ID
Mean Value
Variance
3D Vector
Result
Value

APPENDIX B: Calculating tower crane homogenous transformation matrix based on DH-notation : (a) Schematic; (b) Homogenous transformation matrix; (c) Matlab code.



(a)

$$ {}^{0}T_{3} = \begin{bmatrix} c\theta_1 & -s\theta_1 & 0 & -d_2 s\theta_1 \\ s\theta_1 & c\theta_1 & 0 & d_2 c\theta_1 \\ 0 & 0 & 1 & d_3 \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

(b)

```
1.  clear all;

2.  %Declaring Angles as Matlab Symbolic variables
3.  syms th1 d2 d3;

4.  %Abriviating cosines and sines
5.  c1 = cos(th1);
6.  s1 = sin(th1);

7.  %Declaring transformation matrices for each link
8.  t01 = [c1 -s1 0 0; s1 c1 0 0; 0 0 1 0; 0 0 0 1];
9.  t12 = [-1 0 0 0; 0 0 1 d2; 0 1 0 0; 0 0 0 1];
10. t23 = [-1 0 0 0; 0 0 1 d3; 0 1 0 0; 0 0 0 1];

11. %Computing the t04 symbolically
12. t03 = t01*t12*t23;
13. t03_s = simple(t03);
```

(c)

APPENDIX C: Matlab code for calculating hydraulic crane homogenous transformation matrix symbolically based on DH-notation.

```
clear all;

%Declaring Angles as Matlab Symbolic variables
syms th1 th2 d3 d4;

%Abbreviating cosines and sines
c1 = cos(th1);
s1 = sin(th1);
c2 = cos(th2);
s2 = sin(th2);

%Declaring transformation matrices for each link
t01 = [c1 -s1 0 0; s1 c1 0 0; 0 0 1 0; 0 0 0 1];
t12 = [c2 -s2 0 0; 0 0 -1 0; s2 c2 0 0; 0 0 0 1];
t23 = [1 -0 0 0; 0 0 -1 -d3; 0 1 0 0; 0 0 0 1];
t34 = [1 0 0 0; 0 0 -1 -d4; 0 1 0 0; 0 0 0 1];

%Computing the t04 symbolically
t04 = t01*t12*t23*t34;
t04_s = simple(t04);
```

APPENDIX D: C++ code for *Model3DRigidTreeXSI* Class that is used to model the crane kinematicaly and define the metric.

```
1.  bool Model3DRigidTreeXSI::Satisfied(const MSLVector &x)
2.  {
3.     // Ensure values are within limits
4.     for (int i=0; i<StateDim; i++)
5.     {
6.        if (m_aDynaState[i].isAngular())
7.        {
8.           double a = AngleIntoLimitsRange(x[i],i); // Always above
    LowerState
9.           if (a > UpperState[i]) // In range?
10.             return false;
11.       }
12.       else
13.       {
14.          if ((x[i] > UpperState[i]) || (x[i] < LowerState[i]))
15.             return false;
16.       }
17.    }
18.    if (!ExtendedSatisfied(x))
19.    {
20.       if (m_pOwnerProblem->ActivityRecorder) m_pOwnerProblem-
    >ActivityRecorder->Mark(x,2);
21.       return false;
22.    }
23.    return true;
24. }
25.
26. MSLVector Model3DRigidTreeXSI::StateToConfiguration(const
    MSLVector &x)
27. {
28.    return m_bEvaluateXFormsThroughXSI ?
    StateToConfiguration_XSI(x) : StateToConfiguration_Direct(x);
29. }
30.
31. double Model3DRigidTreeXSI::Metric(const MSLVector &x1, const
    MSLVector &x2)
32. {
33.    double dMetric = 0.0;
34.
35.    for (int i=0; i<StateDim; i++)
36.    {
37.       double weight;
38.       double diff;
39.       if (m_aDynaState[i].isAngular())
40.       {
41.          double a1 = AngleIntoLimitsRange(x1[i],i); // into
    [LowerState,UpperState]
42.          double a2 = AngleIntoLimitsRange(x2[i],i); // into
    [LowerState,UpperState]
```

```
43.             diff = fabs(a2-a1);
44.             if (diff > MATH_PI)
45.                diff = MATH_PI*2 - diff; // Shortest angle
46.             //weight = diff / MATH_PI;//(UpperState[i]-
      LowerState[i]);
47.             //weight = diff / (UpperState[i]-LowerState[i]);
48.          }
49.          else diff = x2[i]-x1[i];
50.
51.          weight = diff / m_aDynaState[i].Speed;
52.          dMetric += weight * weight;
53.       }
54.
55.       return sqrt(dMetric);
56.    }
```

APPENDIX E: C++ source code for the implemented *RRTBiasedConLim* variation.

```
1.  ////////////////// RRTBiasedLimCon Class Implementation
    //////////////////
2.
3.  // This function essentially iterates Extend until it has to stop
4.  // The same action is used for every iteration
5.  bool RRTBiasedLimCon::Connect(const MSLVector &x, MSLTree *t,
    MSLNode *&nn,bool forward)
6.  {
7.     MSLNode *nn_prev = 0;
8.     MSLVector nx,nx_prev;
9.     bool success = false;
10.    double d,d_prev,clock;
11.
12.    MSLNode *n_best = SelectNode(x,t,forward);
13.    MSLVector u_best = SelectInput(n_best-
    >State(),x,nx,success,forward);
14.    if (!success)
15.       return false;
16.
17.    // If a collision-free input was found
18.    // nx gets next state
19.
20.    int steps = 0;
21.
22.    d = P->Metric(nx,x);
23.    d_prev = d;
24.    nx_prev = nx; // Initialize
25.    nn = n_best;
26.    clock = PlannerDeltaT;
27.
28.    double dLast2Goal = P->Metric(n_best->State(),P->GoalState);
29.    double dCurrent2Goal = P->Metric(nx,P->GoalState);
30.    bool bTrackGoalDist = dCurrent2Goal < dLast2Goal;
31.    if (!bTrackGoalDist)
32.    {
33.       // Always succeed in this test
34.       dLast2Goal = 1.0;
35.       dCurrent2Goal = 0.0;
36.    }
37.
38.    while (P->Satisfied(nx) && success && (clock <=
    ConnectTimeLimit) && (d <= d_prev) && (dCurrent2Goal <
    dLast2Goal))
39.    {
40.       SatisfiedCount++;
41.       steps++; // Number of steps made in connecting
42.       nx_prev = nx;
43.       d_prev = d;
44.       nn_prev = nn;
45.
46.       // Select best action each time
47.       //u_best = SelectInput(nn->State(),x,nx,success,forward);
```

```
48.
49.          if (Holonomic)
50.          {
51.             nx = P->Integrate(nx_prev,u_best,PlannerDeltaT);
52.          }
53.          else
54.          {
55.             // Nonholonomic
56.             if (forward)
57.                nx = P->Integrate(nx_prev,u_best,PlannerDeltaT);
58.             else nx = P->Integrate(nx_prev,u_best,-PlannerDeltaT);
59.          }
60.
61.          d = P->Metric(nx,x);
62.          clock += PlannerDeltaT;
63.
64.          if (bTrackGoalDist)
65.          {
66.             dLast2Goal = dCurrent2Goal;
67.             dCurrent2Goal = P->Metric(nx,P->GoalState);
68.          }
69.
70.          // Uncomment the subsequent two lines to
71.          //   make each intermediate node added
72.          //nn = g.new_node(nx_prev); // Make a new node
73.          //g.new_edge(nn_prev,nn,u_best);
74.       }
75.       nn = t->Extend(n_best, nx_prev, u_best, steps*PlannerDeltaT);
76.
77.       return success;
78.    }
```

APPENDIX F: C++ source code for the implemented dynamic RRT algorithm.

```
1.  bool RRTDynamic::StaticPlan()
2.  {
3.     // Create a normal RRT tree. Grow from target to initial.
4.     P->InitialState = GoalState;
5.     P->GoalState = InitialState;
6.
7.     Rrt->NumNodes = m_iInitialNumNodes;
8.     RRTGoalBias *pGoalBias = dynamic_cast<RRTGoalBias*>(Rrt);
9.     if (pGoalBias)
10.       pGoalBias->GoalProb = m_dInitialGoalProbability;
11.
12.    // Initial planning
13.    if (!Rrt->Plan())
14.       return false;
15.
16.    // Success
17.    MoveFromNode = Rrt->GoalNode;
18.    MoveToNode = MoveFromNode->Parent();
19.    return true;
20. }
21.
22. bool RRTDynamic::DynamicReplan(const MSLVector& lastValidState)
23. {
24.    Rrt->NumNodes = m_iReplanNumNodes;
25.    RRTGoalBias *pGoalBias = dynamic_cast<RRTGoalBias*>(Rrt);
26.    if (pGoalBias)
27.       pGoalBias->GoalProb = m_dReplanGoalProbability;
28.
29.    // Mark other nodes in the tree as invalid if detected in
   dynamic obstacles
30.    // from DynaGeom
31.    InvalidateNodes(Rrt->T->Root());
32.
33.    // Edge from lastNode to nextNode is in collision. Mark
   nextNode as invalid.
34.    if (MoveToNode)
35.       SetNodeValidity(MoveToNode, false); // I believe this is a
   valid assumption
36.
37.    // Remove invalid branches
38.    int trimmedNodesCount = TrimTree(Rrt->T);
39.    trimmedNodesCount;
40.
41.    // If the tree became empty, then delete it
42.    if (!Rrt->T->Size())
43.    {
44.       delete Rrt->T;
45.       Rrt->T = NULL;
46.    }
47.
48.    // Grow RRT again to goal if needed
49.    //if (trimmedNodesCount > 0)
```

```
50.      {
51.         P->GoalState = lastValidState; // Grow to last valid state
52.         //P->GoalState = MoveFromNode->State();
53.         Rrt->GoalNode = 0;//MoveFromNode;
54.         Rrt->CumulativePlanningTime = 0; // Reset timer
55.
56.         // Notify implementors of this event
57.         OnTreeTrimmed();
58.
59.         // Replan
60.         if (!Rrt->Plan())
61.         {
62.            MoveFromNode = MoveToNode = 0;
63.            return false;
64.         }
65.
66.         // Setup simulation
67.         ResetSimulation();
68.      }
69.      return true;
70.   }
```

APPENDIX G: C# auto-generated source code for the engineering agent.

```
1.  using System;
2.
3.  namespace MotionPlan
4.  {
5.     public class ExtSatisfy : ExtendedSatisfy.Interface
6.     {
7.        double[] Distances; // Sorted from minimum to maximum
8.        double[][] LoadsPerDistance;
9.        double[][] AnglesPerDistance;
10.
11.       const double Load = 20000;
12.       const int DistanceDOFIndex = 2;
13.       const int ThetaDOFIndex = 1;
14.
15.       public ExtSatisfy()
16.       {
17.          Distances = new double[] {
    108,150,180,210,240,270,300,330 };
18.          LoadsPerDistance = new double[][]
19.          {
20.             new double[]
    {140000,110500,69800,78750,59800,27600,-1,-1,-1,-1,-1,-1,-1,-
    1,-1,-1,-1,-1,-1,-1,-1,},
21.             new double[]
    {109500,104500,91400,75300,59750,47300,38550,28450,16200,-1,-
    1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,},
22.             new double[]
    {84200,79850,73900,59600,50000,42300,36950,28450,23400,19450,11
    350,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,},
23.             new double[] {-1,-1,-1,-
    1,48900,41900,36400,29700,24650,20700,17500,14900,9150,-1,-1,-
    1,-1,-1,-1,-1,-1,},
24.             new double[] {-1,-1,-
    1,50950,43800,38300,33900,30300,25550,21600,18450,15850,13650,1
    1650,7410,-1,-1,-1,-1,-1,-1,},
25.             new double[] {-1,-1,-
    1,41000,35250,31050,27650,24350,22050,20050,18350,16550,14350,1
    2500,10900,9480,6040,-1,-1,-1,-1,},
26.             new double[] {-1,-1,-1,-
    1,34750,30450,27000,24250,21900,19950,18300,16850,14900,13050,1
    1450,10000,8790,7690,4950,-1,-1,},
27.             new double[] {-1,-1,-1,-1,-1,-
    1,25300,22900,20850,19100,17550,16200,15050,13500,11900,10500,9
    260,8150,7170,6280,4060,},
28.          };
29.          AnglesPerDistance = new double[][]
30.          {
31.             new double[] { 22,26,31.5,43,57.5,90,-1,-1,-1,-1,-
    1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1, },
32.             new double[] { 15,17.5,21,28,35,43,52.5,65.5,90,-
    1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1, },
```

```
33.                 new double[] {
    12,14,17,22.5,27.5,33.5,40,47,55.5,67,90,-1,-1,-1,-1,-1,-1,-1,-
    1,-1,-1, },
34.                 new double[] { -1,-1,-
    1,18.5,23,27.5,32.5,38,43.5,50.5,58,69,90,-1,-1,-1,-1,-1,-1,-
    1,-1, },
35.                 new double[] { -1,-
    1,11.5,15.5,19,23,27,31.5,36,41,46.5,52.5,60,70,90,-1,-1,-1,-
    1,-1,-1, },
36.                 new double[] { -1,-
    1,10,13,16.5,19.5,23,27,30.5,34.5,39,43.5,48.5,54.5,61,71,90,-
    1,-1,-1,-1, },
37.                 new double[] { -1,-1,-
    1,11,14,17,20,23,26.5,30,33.5,37,41,45.5,50.5,55.5,62,71.5,90,-
    1,-1, },
38.                 new double[] { -1,-1,-1,-1,-
    1,14.5,17.5,20,23,26,29,32.5,35.5,39.5,43,47,51.5,57,63,71.5,90
    , },
39.             };
40.         }
41.         #region Interface Members
42.         public bool Satisfy(double[] state)
43.         {
44.             double distance = state[DistanceDOFIndex];
45.             double angle = (state[ThetaDOFIndex] / Math.PI) *
    180.0;
46.             int DistanceGroup = -1;
47.             for (int i = 0; i < Distances.Length; i++)
48.             {
49.                 if (distance <= Distances[i])
50.                 {
51.                     DistanceGroup = i;
52.                     break;
53.                 }
54.             }
55.             if (DistanceGroup == -1)
56.                 return false;
57.             double[] Loads = LoadsPerDistance[DistanceGroup];
58.             int LoadGroup = -1;
59.             int MinLoadGroup = -1;
60.             for (int i = Loads.Length - 1; i >= 0; i--)
61.             {
62.                 if (Loads[i] > -1)
63.                 {
64.                     if (MinLoadGroup == -1)
65.                         MinLoadGroup = i;
66.                     if (Load <= Loads[i])
67.                     {
68.                         LoadGroup = i;
69.                         break;
70.                     }
71.                 }
72.             }
73.             if (LoadGroup == -1)
74.             {
75.                 if (Load < Loads[MinLoadGroup])
76.                     LoadGroup = MinLoadGroup;
```

```
77.                 else return false;
78.             }
79.             double MinAngle = 0.0;
80.             double MaxAngle =
    AnglesPerDistance[DistanceGroup][LoadGroup];
81.             return (angle >= MinAngle) && (angle <= MaxAngle);
82.         }
83.         #endregion
84.     }
85. }
```

APPENDIX H: Verbose logging for the solving process: This log shows the main steps in the process such as accessing and analyzing different components of the problem including: the geometry of the crane and the obstacles, kinematic properties of the crane and the engineering agent code. It also logs the planning results in summery.

```
1.  // VERBOSE : MotionPlan_RealTimeDRRT_OnInit called
2.  // VERBOSE : MotionPlan_RealTimeDRRT_Solve_OnClicked called
3.  // INFO : Ending last dynamic motion plan session before
    initiating new one
4.  // INFO : Dynamic motion plan session ended
5.  // INFO : Grabbing robot root at: robot
6.  // INFO : Robot root found: robot
7.  // INFO : Flattenning robot hierarchy
8.  // INFO : Grabbing obstacles root at: obstacles
9.  // INFO : Obstacles root was found
10. // INFO : Generating internal geometry representation
11. // INFO : Robot geometry triangulation
12. // INFO : Robot geometry robot has 12 triangles
13. // INFO : Robot geometry rev02 has 12 triangles
14. // INFO : Robot geometry prism03 has 12 triangles
15. // INFO : Robot geometry fixAxis has 6 triangles
16. // INFO : Robot geometry prism04 has 12 triangles
17. // INFO : Obstacles geometry triangulation
18. // INFO : Total obstacles geometry triangles: 6360
19. // INFO : Generating internal model representation
20. // INFO : Dynamic channel (00): robot.roty
21. // INFO : Dynamic channel (01): rev02.rotx
22. // INFO : Dynamic channel (02): prism03.posy
23. // INFO : Dynamic channel (03): prism04.posx
24. // INFO : Grabbing extended satisfy C# script at:
    robot.ext_constraints_cs.text
25. // INFO : Extended satisfy C# script was not found
26. // INFO : Grabbing extended satisfy C++ script at:
    robot.ext_constraints_cpp.text
27. // INFO : Extended satisfy C++ script was not found
28. // INFO : Will use XSI transform evaluation (XSI
    constraints/expressions supported)
29. // INFO : Gathering initial and goal states
30. // INFO : Generating problem description
31. // INFO : Generating planner: RRTLimGoalBias
32. // INFO : Generating high-level XSI dynamic planner
33. // INFO : Random seed: 0
34. // INFO : Activating visualizer
35. // INFO : Visualizer successfully activated
36. // INFO : Static Plan phase (phase 1)
37. // INFO : Resetting simulation internal parameters
38. // INFO : Done Static Plan phase (phase 1)
```

```
39.   // INFO : Phase 2 planner was avoided because phase 1 succeeded
40.   // INFO : Planner succeeded. Removing robot keyframes for direct
      planner control
41.   // INFO : Planning Time: 8.95801s
42.   // Successful Path Found
43.   //
44.   // INFO : Path result: 1
```

APPENDIX I: Detailed results from the Scene Debugger. These results show the performance of the different tasks in the motion planning process.

Scope	Net Total	Executing	# of requests
0.8% Drawing	-	101	5,125
0.4% Draw Primitive	-	47	953
0.4% Transform	-	44	2,085
74.4% Tasks	-	9,389	1
74.4% MotionPlan (task)	-	9,389	1
0.5% OpenGL Render	-	57	1
0.5% OpenGL Render View B	-	57	1
23.1% Animation Operators	-	2,916	54,416
17.8% crane_sim.ropeRoll.kine.global.IndpCns_E	-	2,248	1,554
0.4% crane_sim.ropeRoll.kine.local.PoseCompensatorOp	-	49	1,554
0.4% prism04.kine.global.ParentPoseCns_E	-	48	3,104
0.4% rev02.kine.global.ParentPoseCns_E	-	47	3,106
0.3% rev02.kine.local.PoseCompensatorOp	-	40	3,106
0.3% prism03.kine.global.ParentPoseCns_E	-	34	3,106
0.3% prism03.kine.local.PoseCompensatorOp	-	34	3,106
0.2% crane_sim.prjTip.kine.global.ParentPoseCns_E	-	31	1,554
0.2% crane_sim.ropeRoll.kine.objclscns.ObjectToClusterCnsDefiner	-	30	1,554
0.2% fixAxis.kine.global.IndpCns_E	-	24	1,554
0.2% robot.kine.global.ParentPoseCns_E	-	23	1,555
0.2% crane_sim.prismA.kine.global.ParentPoseCns_E	-	20	1,554
0.2% prism04.kine.local.PoseCompensatorOp	-	20	1,554
1.1% Other Operators	-	144	10,879
0.4% crane_sim.prismB.kine.local.Expression	-	48	1,554
0.2% crane_sim.prismC.kine.local.Expression	-	25	1,554
0.2% crane_sim.prismD.kine.local.Expression	-	24	1,554

Scope	Net Total	Executing	# of requests
98.6% MotionPlan (task)	12,440	9,389	1
22.6% prism04.kine.local.PoseCompensatorOp	2,851	20	1,553
22.4% prism04.kine.global.ParentPoseCns_E	2,830	18	1,553
22.3% prism04.kine.local.ParentPoseAndPoseCns_D	2,811	18	1,553
22.1% fixAxis.kine.global.IndpCns_E	2,793	24	1,553
21.9% fixAxis.kine.dircns.PosePoseCns_D	2,768	17	1,553
21.5% crane_sim.prjTip.kine.global.ParentPoseCns_E	2,708	31	1,553
21.2% crane_sim.prjTip.kine.local.Expression	2,669	17	1,553
21.0% crane_sim.ropeRoll.kine.local.PoseCompensatorOp	2,652	49	1,553

```
|  |  |  |  \--+ 20.6% crane_sim.ropeRoll.kine.global.IndpCns_E  2,602              2,247              1,553
|  |  |  |     \--+  2.8% crane_sim.ropeRoll.kine.objclscns.ObjectToClusterCnsDefiner35530              1,553
|  |  |  |        |--+  2.4% crane_sim.prismD.kine.global.ParentPoseCns_E301         19                 1,553
|  |  |  |        |  |--+  2.0% crane_sim.prismD.kine.local.ParentPoseAndPoseCns_D25711                1,553
|  |  |  |        |  |  \--+  2.0% crane_sim.prismC.kine.global.ParentPoseCns_E246   18                 1,553
|  |  |  |        |  |     |--+  1.6% crane_sim.prismC.kine.local.ParentPoseAndPoseCns_D20210          1,553
|  |  |  |        |  |     |  \--+  1.5% crane_sim.prismB.kine.global.ParentPoseCns_E19118             1,553
|  |  |  |        |  |     |     |--+  0.9% crane_sim.prismB.kine.local.ParentPoseAndPoseCns_D11110    1,553
|  |  |  |        |  |     |     |  \--+  0.8% crane_sim.prismA.kine.global.ParentPoseCns_E10020       1,553
|  |  |  |        |  |     |     |     |--+  0.5% crane_sim.prismA.kine.local.ParentPoseAndPoseCns_D5811 1,553
|  |  |  |        |  |     |     |     |  \--+  0.4% crane_sim.hud.kine.global.ParentPoseCns_E4717     1,553
|  |  |  |        |  |     |     |     |     \---  0.2% crane_sim.hud.kine.local.Expression2911        1,553
|  |  |  |        |  |     |     |     \---  0.2% crane_sim.prismA.kine.local.Expression219          1,553
|  |  |  |        |  |     |     \---  0.5% crane_sim.prismB.kine.local.Expression61 48                1,553
|  |  |  |        |  |     \---  0.2% crane_sim.prismC.kine.local.Expression25       25                 1,553
|  |  |  |        |  \---  0.2% crane_sim.prismD.kine.local.Expression24            24                 1,553
|  |  |  |        \---  0.2% crane_sim.ropeRoll.kine.global.ParentPoseCns_E23       11                 1,553
|  |  |  \--+  0.3% fixAxis.kine.global.ParentPoseCns_E                42             11                 1,553
|  |  |     \---  0.2% fixAxis.kine.local.ParentPoseAndPoseCns_D       30             11                 1,553
|  |--+  0.8% rev02.kine.local.PoseCompensatorOp                       95             27                 1,553
|  |  \--+  0.5% rev02.kine.global.ParentPoseCns_E                     68             26                 1,553
|  |     \--+  0.3% rev02.kine.local.ParentPoseAndPoseCns_D            41             18                 1,553
|  |        \---  0.2% robot.kine.global.ParentPoseCns_E               22             22                 1,552
|  |--+  0.6% prism03.kine.local.PoseCompensatorOp                     74             21                 1,553
|  |  \--+  0.4% prism03.kine.global.ParentPoseCns_E                   52             15                 1,553
|  |     \--+  0.3% prism03.kine.local.ParentPoseAndPoseCns_D          36             15                 1,553
|  |        \---  0.2% rev02.kine.global.ParentPoseCns_E               20             20                 1,552
|  \---  0.2% prism04.kine.global.ParentPoseCns_E                      30             30                 1,550
\---  1.4% OpenGL Render View B                                       174            57                 1
```

APPENDIX J: Reasons that cause capacity reduction for hydraulic cranes.

1. Failure to block/crib under the outrigger pads when poor ground conditions cannot support the total weight of the crane and load. Proper and improper cribbing is shown in Figure J.1.
2. Failure to extend the outriggers fully and use them following the manufacturer's instruction.
3. Failure to note overhead obstructions, such as overpasses and power lines.
4. Failure to level the crane. Leveling the crane cannot be overemphasized. Cranes must be set up as per manufacturer's instruction with the outriggers fully extended and the crane leveled. Crane capacity is lost when the crane is out of level by only a few degrees. Table shows the values of capacity reduction caused by out of level. Most cranes have levels mounted on them, but the levels are not always accurate.

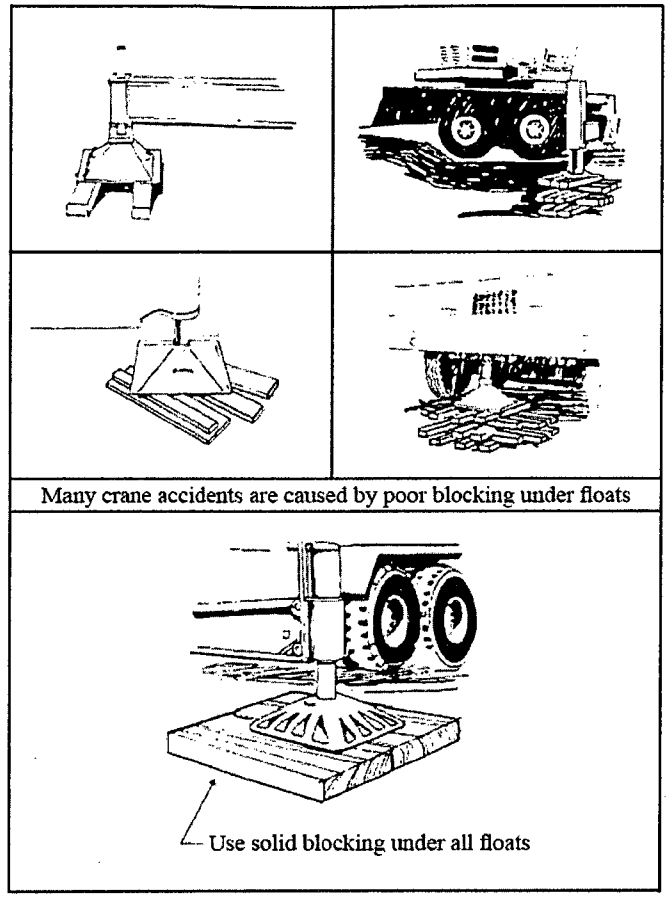


Figure J.1: Proper and improper cribbing of outrigger pads (Integrated Publishing, 2009)

Table J.1: Sample table for capacity reduction caused by crane out of level (Integrated Publishing, 2009)

Boom Length and Lift Radius	Chart Capacity Lost When Crane Out of Level By		
	1°	2°	3°
Short Boom, Minimum Radius	10%	20%	30%
Short Boom, Maximum Radius	8%	15%	20%
Long Boom, Minimum Radius	30%	41%	50%
Long Boom, Maximum Radius	5%	19%	15%

APPENDIX K: List of Publications

Articles accepted / submitted to refereed journals

Homam AlBahnassi and Amin Hammad (submitted). Framework for Real-Time Motion Planning and Simulation of Heavy Construction Equipment, Journal of Computing in Civil Engineering, ASCE.

Homam AlBahnassi and Amin Hammad (submitted). Sampling-Based Algorithm for Motion Planning of Hydraulic Cranes in Dynamic Environments, Journal of Computing in Civil Engineering, ASCE.

Cheng Zhang, Amin Hammad and Homam AlBahnassi, Collaborative Multi-Agent Systems for Construction Equipment Based on Real-Time Field Data Capturing, Journal of Information Technology in Construction (ITcon), Vol. 14, Special Issue Next Generation Construction IT: Technology Foresight, Future Studies, Roadmapping, and Scenario Planning, pp. 204-228, http://www.itcon.org/2009/16.

Articles accepted / submitted to refereed conferences

Homam AlBahnassi, Amin Hammad and Cheng Zhang, Accurate Heavy Equipment Motion Planning Considering Local and Global Constraints, In Proceedings of

2nd International/ 8th Construction Specialty Conference. St. John's, Newfoundland and Labrador, May, 2009.

Cheng Zhang, Homam AlBahnassi and Amin Hammad, Improving Construction Safety through Real-Time Motion Planning of Cranes, The International Conference on Computing in Civil and Building Engineering (ISCCBE) - 2010, June 30-July 2, 2010, Nottingham, UK.

Cheng Zhang, Amin Hammad and Homam AlBahnassi, Path Re-planning of Cranes Using Real-Time Location System, In Proceedings of 2009 26th International Symposium on Automation and Robotics in Construction (ISARC 2009), Austin, Texas, USA.

Cheng Zhang, Amin Hammad and Homam AlBahnassi, Collaborative Multi-Agent System for Supporting Construction Equipment, In Proceedings of 2nd International/8th Construction Specialty Conference. St. John's, Newfoundland and Labrador, May, 2009.