Dynamic Fixture Planning in Virtual Environments

By

Xiu Mei Kang

A thesis submitted to

the Faculty of Graduate Studies

in partial fulfilment of

the requirements for the degree of

Doctor of Philosophy

Department of Mechanical and Manufacturing Engineering

Faculty of Engineering

University of Manitoba

Winnipeg, Manitoba

June 2010

Abstract

In today's highly dynamic manufacturing environments, small volume customized products lead to frequent parts change which increases the process complexity. A rapid response to the change is crucial for an enterprise to remain competitive. New emerging technologies are developed and employed to address such challenge. As a promising technology, Virtual Reality (VR) has gained great attention in various industrial areas including product design, virtual prototyping, virtual assembly, manufacturing simulation, and facility planning.

Computer-aided fixture planning (CAFP) is an essential part of Computer-aided design and manufacturing (CAD/CAM) integration. Proper fixture planning can dramatically reduce the manufacturing cost, the lead-time, and labor skill requirements in product manufacturing. However, fixture planning is a highly experience-based activity. Due to the extreme diversity and complexity of manufacturing workpieces and processes, there are not many fixture planning tools available for industry applications. Moreover, existing CAFP methods rarely consider integrating fixture environmental factors into fixture planning. Automatic fixture planning using VR can provide a viable way for industries.

This thesis develops automated approaches to fixture planning in a virtual environment (VE). It intends to address two important issues: automatic algorithms for fixture planning, and the VE to support high fidelity evaluation of fixture planning. The system consists of three parts including fixture assembly planning, feasibility analysis of assembly

tools, and motion planning for workpiece loading and unloading. The virtual fixture planning system provides the fixture designer a tool for fixture planning and evaluation. Geometrical algorithms are developed to facilitate the automatic reasoning.

A Web-based VE for fixture planning is implemented. It supports geographically distributed collaborations of the fixture design and applications. The three-dimensional (3D) model visualization enables the fixture simulation and validation effectively to investigate existing problems. Approaches to construct desktop-based large VEs are also investigated. Cell segmentation methods and dynamic loading strategies are investigated to improve the rendering performance. Case studies of virtual building navigation and product assembly simulations are conducted.

The developed algorithms can successfully generate the assembly plan, validate the assembly tools, and generate moving paths for fixture design and applications. The VE is intuitive and sufficient to support fixture planning, as well as other virtual design and manufacturing tasks.

Acknowledgments

I would like to thank my advisor, Dr. Qingjin Peng, for his constant support and guidance in my research. Without him, it would be probably impossible for me to complete this thesis. I deeply appreciate his encouragement and belief in me. I learned a lot from Dr. Peng both in research and positive living attitude.

My appreciation is sincerely expressed to my committee members, Dr. Subramaniam Balakrishnan and Dr. Pourang Polad Irani, for their invaluable guidance and comments during my research. Their unconditional support leads me to the end.

I am very grateful to my current and former colleagues working together in the Virtual Manufacturing Centre (VMC), names of a few, Dr. Chulho Chung, Dr. Chunsheng Yu, Mahmud Ahsan, Qiu Niu, Tong Zhang, Tingting Zhao, and Long Zhao, for their help in my thesis development. My gratitude also goes to my friends without exception for the happy hours we spent together.

I would like to express my thanks to Natural Sciences and Engineering Research Council of Canada (NSERC), the Faculty of Engineering, and the Faculty of Graduate Studies at the University of Manitoba for their financial support for my research. I also wish to express my deepest gratitude to those generous people for providing the Berdie & Irvin Cohen Fellowship in Engineering, and Edward R. Toporeck Graduate Fellowship in En-

gineering. These supports make it possible for me to concentrate on my research and study.

My deep appreciation goes to my husband, Ming Song, and my son, Jack Song. I am grateful to my husband's encouragement, trust, and support without any doubt and complaint. Without him, it is impossible for me to obtain this degree. I would also want to thank my beloved parents and my brothers and my sister for their support and love to me.

I would like to express my deep appreciation to the following publishers for allowing me to include the copyrighted materials in the thesis:

Bentham Science Publishers Ltd.: http://www.bentham.org/

Original source of publications from Bentham Science Publishers Ltd.:

Recent Patents on Mechanical Engineering: http://www.bentham.org/meng/EBM.htm

CAD Solutions, LLC: http://www.cadanda.com/

Original source of publications from CAD Solutions, LLC:

Computer-aided design and applications: http://www.cadanda.com/

CIRP: http://www.cirp.net/

Original source of publications from CIRP:

The 16th CIRP International Design Seminar: design & innovation for sustainable society

InderScience Publisher: http://www.inderscience.com/

Original source of publications from InderScience Publisher:

Contents

List of Tables

List of Figures

List of Copyrighted Material

Use of Kang, X., Q. Peng. 2010. Data integration from product design to assembly planning in a collaborative environment. *International Journal of Manufacturing Research* 5(1): 120-137.

Use of Kang, X., Q. Peng. 2009. Recent Research on Computer-Aided Fixture Planning. *Recent Patents on Mechanical Engineering* 2(1): 8-18.

Use of Peng, Q.-J.., X.-M. Kang, T.-T. Zhao. 2009. Effective VR-based building navigation using dynamic loading and path optimization. *International Journal of Automation and Computing* 6(4): 335-343. DOI: 10.1007/s11633-009-0335-9.

Use of Kang, X., Q. Peng. 2008. Fixture assembly planning in a Web-based collaborative environment. *International Journal of Internet Manufacturing and Service* 1(2): 176-193.

Use of Kang, X., Q. Peng. 2008. Fixture feasibility: methods and techniques for fixture planning. *Computer-aided design and applications* 5(1-4): 424-433.

Use of Kang, X. and Q. Peng. 2006. The improvement of VE performance using cell segmentation. *Proceedings of the 16th CIRP International Design Seminar*. July 16-19, Kananaskis, Alberta, Canada.

Use of Kang, X., Q. Peng. 2008. Tool feasibility analysis for fixture assembly planning. *Journal of Manufacturing Science and Engineering, Transactions of the ASME* 130(4): 0410101-9.

Use of Kang, X., Q. Peng. 2008. Computer-aided fixture planning: a review. *Proceedings of the ASME 2008 Design Engineering Technical Conferences and Computers and Information in Engineering Conferences (IDETC/CIE2008)* 5: 209-218. August 3-6, New York City, NY, USA.

Use of Kang, X., Q. Peng. 2007. Analysis of tool accessibility in fixture setup planning, Proceedings of the *ASME 2007 Design Engineering Technical Conferences and Computers and Information in Engineering Conferences (IDETC/CIE2007)* 4: 1007-16. September 4-7, Las Vegas, Nevada, USA.

List of Symbols

Symbol	Description
$GAC^{d}_{(R)}$	Global accessibility cone with depth of a truncated half-line
e_a	Height of the tool body
e_b	Height of the tool head
e_e	Height of the tool extension
e_f	Tool adjustment in the fastener removal direction (y-axis)
h	Height of the tool handle
H_j	Location of the tool joint from the top of a fastener
h_n	Height of the tool end
l_e	Length of the tool handle
l_f	Tool displacement in the fastener removal direction (y-axis) during a tool application
l_n	Length of the tool end
R	The maximal radius of the tool occupied space centered at tool joint
r_a	Radius of the tool body
r_b	Radius of the tool head
r_e	Radius of the tool extension
w	Width of the tool handle

Symbol	Description
w_n	Width of the tool end
α_{min}	Minimal tool access angle
α_{max}	Maximal tool access angle
β	Minimal tool application angle of the tool body from the top of a fastener
φ	Colatitude angles
θ	Longitude angle
$\hat{\boldsymbol{T}}(\theta, \varphi)$	Unit vector on a pixel (θ,φ)
M	Set of fasteners in a fixture, {1, 2, …, m}
N	Set of fixture assembly elements, {1, 2, …, n}
FP	Fastener-element connectivity matrix (m x n)
FA	Fastener accessibility matrix (m x n)
PA	Element accessibility matrix (n x n)
DL	Element disassembly level matrix (n x n)
D^3C	Digitalized disassembly directionality chart
Δi	Topological disassemblability of element i
λi	Topological accessibility of fastener i
DPI	A disassembly priority index
$G_{DP}(dpi)$	An element set whose DPI equals dpi
L	A one step moving distance of an object
S	Spherical accessibility matrix
$g(n)$	The cost from start to node n

Symbol	Description
h(n)	The heuristic estimate cost from node *n* to goal
f(n)	The total cost

Chapter 1

Introduction

1.1 Background

In today's highly dynamic manufacturing environments, small volume customized products lead to frequent parts change which increases parts processing complexity. A rapid response to the change is crucial for an enterprise to survive. Moreover, due to the fierce global competition, manufacturing companies are facing the challenges of improving productivity and quality, reducing production costs, shortening time-to-market, and resolving environmental issues. They have to employ new emerging technologies to remain competitive. Virtual Reality (VR) is an enabling technology to simulate a product's performance and production process in a global collaborative environment before a product is actually made.

Fixtures are used to locate, hold, and support workpieces in manufacturing operations such as machining, inspection, and assembly. They play an important role in manufacturing. Designing and fabricating fixtures could cost up to 10-20% of the total manufactur-

ing system cost (Bi and Zhang, 2001). A traditional fixture, called dedicated fixture, is designed to hold a specific workpiece. Usually, it is time-consuming and costly to design and manufacture dedicated fixtures because of the fixtures' tight tolerance and restricted machining operations. Flexible and adaptable fixtures can dramatically reduce the fixturing cost and lead-time since they are reusable (Hargrove and Kusiak, 1994). Using flexible and adaptable fixtures, one can expect as much as 80% reduction in the fixture cost. Modular fixture, one type of flexible fixture, is a set of ready-made, re-usable, standard component and combination unit. The set includes base plates, supports, locators, clamps, and accessories. Modular fixtures provide many fixture configurations for different workpieces using a variety of fixture element combinations.

Fixture planning determines fixture configuration, fixture components, and assembly according to a part design and process requirements. Proper fixture planning can dramatically reduce the manufacturing cost, the lead-time, and labor skill requirements in product manufacturing. With the aid of computer based techniques, Computer-aided fixture planning (CAFP) improves the efficiency and feasibility of fixture design. Also, the integration of fixture planning with computer-aided design and manufacturing (CAD/CAM) provides an overall optimal solution for product design and manufacture.

Much work related to fixture planning has been reported in the past decades. However, fixture planning is a highly experience-based activity. There are not many fixture planning tools available for manufacturing applications due to the extreme diversity and complexity of workpieces and processes. New manufacturing technologies, such as computer-integrated manufacturing, flexible manufacturing systems, lean manufacturing, agile

manufacturing and internet-based manufacturing, bring new challenges to fixture planning in terms of the planning theory, planning reliability and planning efficiency. Moreover, the performance of a fixture system is affected by its surrounding components, called environmental factors, during the fixture operations and machining process. Existing CAFP methods rarely consider integrating environmental factors into fixture planning, which may have a great impact on fixture feasibility. Although optimization of the manufacturing processing, such as process planning and scheduling, has been intensely investigated, to some extent, the planning automation in fixture operations has been neglected, especially for highly customized products.

The economic globalization brings challenges to the collaboration of product development. Such collaboration is not restricted to within an enterprise because many companies outsource their jobs to down-stream factories located in developing countries to save cost. Its effectiveness and success rely on data, information, and knowledge sharing under geographically distributed working environments. Since the last decade, the Web-based environment has been widely used as an enabling platform for the development of collaborative applications in product design and manufacturing (Wang et al., 2002). A Web-based system for automatic fixture planning provides a new design tool in VEs to support geographically distributed collaborations. Meanwhile, the three-dimensional (3D) model visualization enables the simulation and virtual validation of various analysis results to help engineers easily investigate existing problems. The integration of fixture planning with product design in Web-based virtual environments provides a cost-effective solution for the product life cycle management.

1.2 Research motivation

Traditional CAD-based fixture planning methods for mass-production provide platforms for the generation, selection, positioning and assembly of the fixture elements. However, it is difficult to cooperate and share information under these systems because the data structures of CAD systems are different. The function of design feasibility analysis of CAD systems is also limited. Moreover, existing CAFP methods rarely consider integrating fixture environmental factors into fixture planning. Automated approaches for fixture planning are essential to deal with the dynamic requirements of customers in the rapidly changing manufacturing environments. A Web-based system is required for information sharing between fixture designers and other engineers in a distributed environment. Automatic fixture planning using VR technology can provide a viable way for industrial practice.

VR is an emerging technology being used in diverse research fields and industries. Over the last decades, substantial research has been conducted to explore the application of VR in computer games, educational training, industrial design and product assembly, manufacturing simulation, and facility planning, etc. Despite the successful use of VR in various industries, less attention has been paid to automatic fixture planning using VR. VR has a great potential to address the challenge of fixture planning in dynamic manufacturing environments. Dynamic fixture planning considers changeable environments of the fixture application, including the operation space for the fixture assembly and disassembly, workpiece accessibility in the fixture related to the machine used, and tools' feasibil-

ity used in the fixture operation. This research aims to fill such a gap to develop automatic approaches of fixture planning under VR.

1.3 Research objectives

The research objectives of this thesis are to develop automatic approaches for fixture planning in virtual manufacturing environments. It intends to address two important issues: automatic algorithms for fixture planning, and the construction and performance improvement of VEs to support high fidelity simulation of fixtures. The automatic fixture planning includes three parts: fixture assembly planning, feasibility analysis of assembly tools, and motion planning for fixture loading and unloading. It will integrate environmental factors into the automatic fixture planning to meet the dynamic challenge. The proposed VE should be generic, flexible and dynamic for fixture planning systems, as well as for other virtual design and manufacturing tasks. It should also provide a distributed and collaborative environment. Specific tasks of this research are as follows:

1. Develop a fixture assembly planning system to meet dynamic manufacturing system requirements.

2. Enable the fixture environmental elements in fixture planning to reflect the dynamic change of manufacturing environments.

3. Develop approaches to the feasibility analysis of assembly tools, which can automatically select and evaluate assembly tools used in fixture assembly.

4. Establish an automatic geometrical reasoning algorithm for fixture assembly sequence planning based on geometrical constraints, assembly tools, and the requirements of VR data format and structure.

5. Develop a motion planning algorithm for automated fixture loading and unloading.

6. Implement a Web-based virtual fixture planning system to provide an easy access for distributed users. An intuitive user interface will be developed based on VR techniques for fixture modeling and visualization.

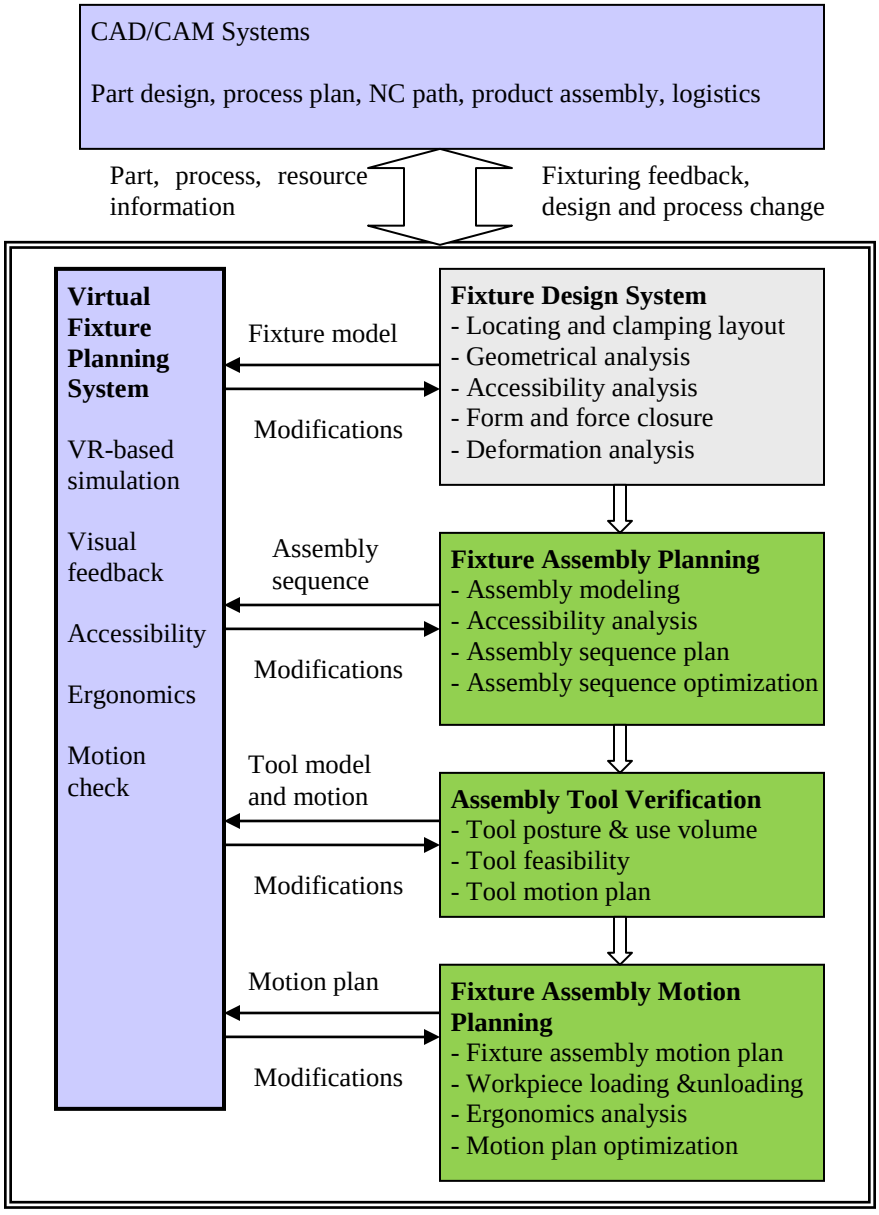
7. Investigate approaches to construct large-scale VEs. Case studies will be conducted.

1.4 Overview of the proposed research

This thesis explores automatic approaches of fixture planning, and improvements of VEs' construction and performance. A Web-based virtual fixture planning system is developed. Figure 1.1 shows the structure of the proposed virtual fixture planning system. The system consists of three subsystems including fixture assembly planning, assembly tool verification, and motion planning in fixture loading and unloading. Fixture assembly planning subsystem automatically generates the fixture assembly sequence with respect to geometry and functional constraints. The assembly tools verification subsystem chooses and evaluates the feasibility of assembly tools. The motion planning subsystem generates the motion plan for a workpiece during its loading and unloading under complex manu-

facturing environments. Environmental factors in fixture assembly are integrated into the algorithms to address the dynamically changing manufacturing environment.

Figure 1.1 The architecture of proposed virtual fixture planning system



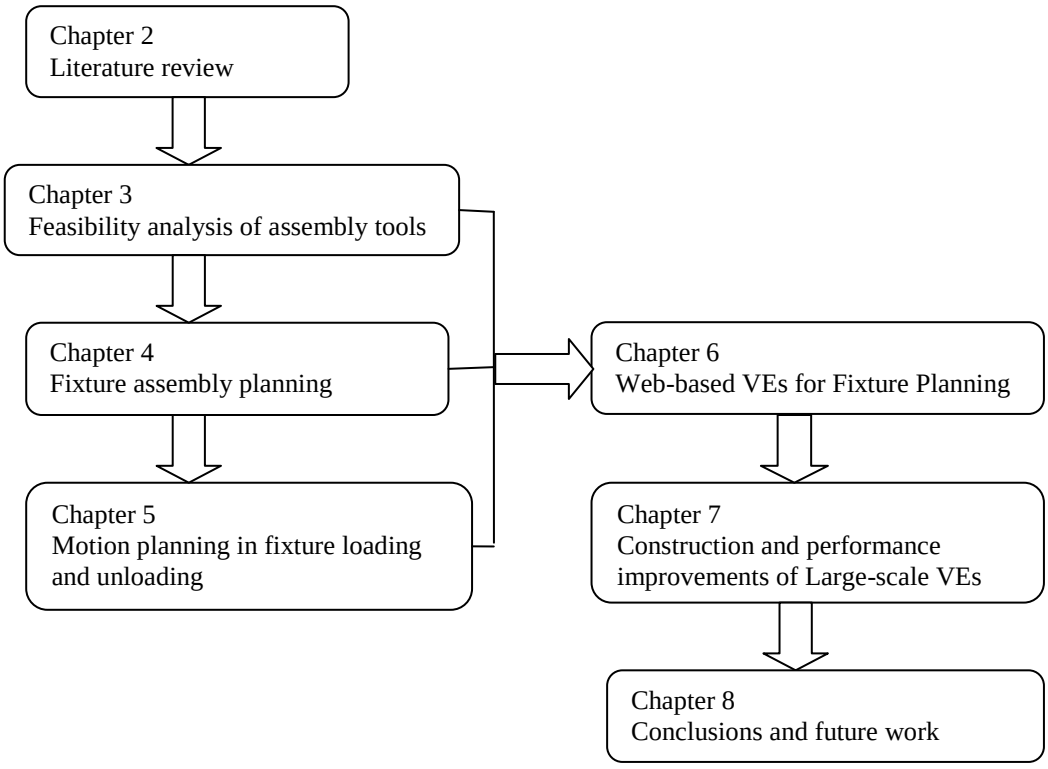
The input data for the proposed system is a fixture design from a CAD/CAM system. The data are used to generate fixture configurations, evaluate the fixturing feasibility, and verify the assembly possibility in the fixturing environment. The fixturing feedback is sent back to the CAD/CAM system for further design and process modifications.

Besides the Web-based VE, desktop-based VEs are also investigated including the construction of large VEs and their performance improvement for building navigations and product assembly simulation.

1.5 Thesis structure

The organization of the thesis is shown in Figure 1.2.

Figure 1.2 Structure of the thesis



Chapter two looks at the background and previous research concerning CAFP. The issues in fixture planning and related approaches to fixture planning are reviewed. The implementation environments for CAFP systems including CAD-based systems, VR-based systems, and Web-based systems are discussed. Limitations of current research and promising research areas are identified.

Chapter three describes the feasibility analysis of assembly tools in fixture assembly planning. Following a close look at the literature related to tool assessment, the modeling of assembly tools and a new $GAC^{d}_{(R)}$ are defined. A $GAC^{d}_{(R)}$ represents obstacles around a tool within the effective tool maximum length. The feasibility analysis of assembly tools are executed based on the tools' use volume and $GAC^{d}_{(R)}$ information. A tool's use volume refers to the minimum free space in a subassembly to apply the tool (Wilson, 1998).

Chapter four introduces a fastener-based fixture assembly planning method which is based on the analysis of topological disassemblability, feasibility of assembly tool, and fixture functional constraints. Two steps to generate an assembly sequence plan are described and two examples of fixture assembly planning are illustrated.

Chapter five presents a new accessibility-based A Star (A*) algorithm for motion planning in fixture loading and unloading. The construction of a spherical accessibility matrix is demonstrated to identify accessible moving directions of a workpiece based on geometrical reasoning. Then A* algorithm is described to search for the optimal moving path based on a spherical accessibility matrix.

Chapter six describes the implementation of a Web-based VE and the data integration from 3D CAD models to the VR simulation for assembly planning and 3D visualization. It discusses techniques of extracting geometrical data from virtual reality modeling language (VRML) files, the reconstruction of 3D models and dynamic animation in the VE.

Chapter seven demonstrates the construction of large-scale VEs and dynamic loading strategies for performance improvements in simulation. Two case studies, including building navigation and product assembly simulation, are conducted to validate the effectiveness of the proposed methods.

Chapter eight summarizes the conducted research and identifies the contributions of this thesis, followed by the recommendation for further research.

Chapter 2

Literature review

Fixture planning is a complex activity affected by the diversity of workpieces and environmental factors during machining, assembly, and inspection. This chapter introduces CAFP and the related issues of CAFP. It surveys existing research on fixture assembly planning, assembly tools' feasibility analysis, and fixture loading and unloading. The implementation environments for CAFP systems including CAD-based systems, Web-based systems, and VR-based systems are discussed. Limitations of existing research and research needs for fixture assembly planning and fixture loading and unloading are identified.

2.1 Introduction to CAFP

Fixture planning determines precise locating and rigid clamping of workpieces according to workpieces' design and process requirements. A locating planning chooses surfaces on the workpiece for a fixture to hold the workpiece reliably. Locating surfaces are classified as planes, pin-holes and external profiles. Commonly used fixture locating methods include: (1) 3-2-1 point locating for prismatic parts. (2) Pin-hole locating for general parts

with internal cylindrical features. (3) V-block locating for parts with external cylindrical features, which may use a wide V-block or two short V-pads to hold the workpiece.

The clamping methods can be summarized as top clamping and side clamping. It is usually used to restrict a workpiece's movement and keep it stable during processing (Trappey and Liu, 1990). The clamping planning determines clamping surfaces and points on the workpiece and the clamping components, the magnitude of clamping forces, and the clamping sequence. Fixture assembly planning determines the fixture assembly sequence to ensure an interference-free assembly process and the ease of workpieces loading and unloading.

Figure 2.1 The 3-2-1 locating method for a gearbox body

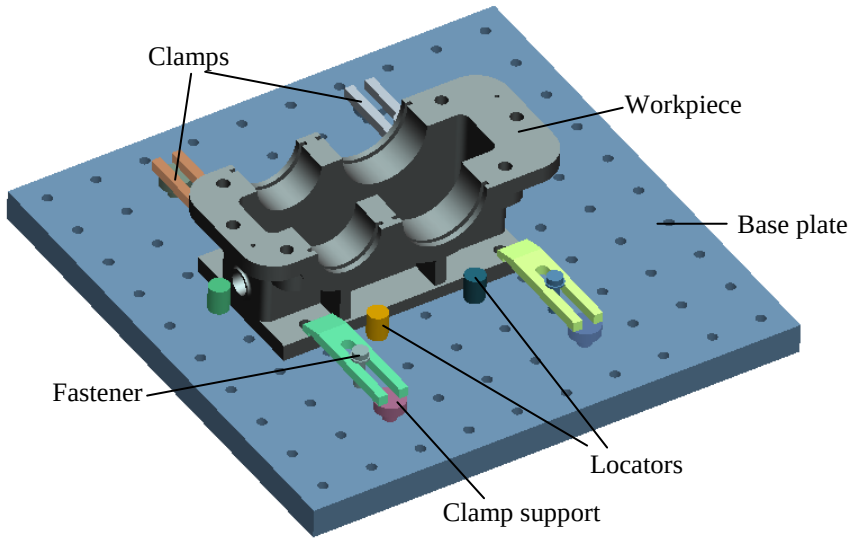


Figure 2.1 illustrates a 3-2-1 point locating fixture for a gearbox body which is a prismatic workpiece. The workpiece is located by three perpendicular locating planes. The

bottom plane of the workpiece forms the primary locating plane. The plane contacting two locators is the secondary locating plane. The left plane with a hole is the tertiary locating plane. Four vertical clamps are applied on two sides of the workpiece. The features on the top surface of the workpiece are to be machined.

The design of a fixture is constrained by the workpiece and its machining requirements, such as the geometry, material, dimension and tolerance, process plan, and tool path. The main design constraints in fixture planning are summarized as follows:

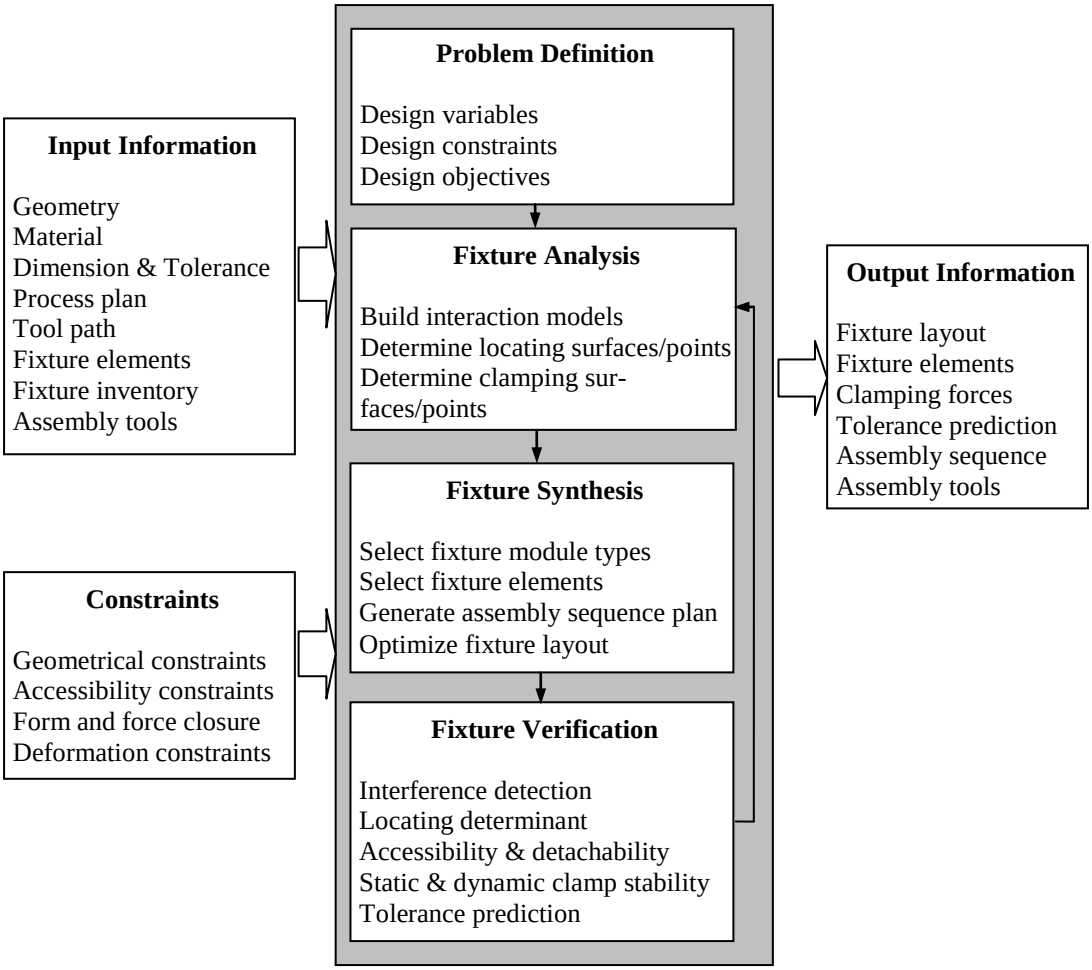
(1) Geometrical constraints: the location of a workpiece should be ensured to keep the machining accuracy of the workpiece within the design specifications.

(2) Accessibility constraints: there should be no interference among fixture components, workpieces, and machining tools during assembly and machining. In addition, it should be easy to load and unload the workpiece.

(3) Force constraints: the fixture should be strong enough to resist the forces and moments produced by clamps and machining tools. A minimum clamp force should be specified for the workpiece stability.

(4) Deformation constraints: the stiffness of a fixture system should be sufficient to keep the workpiece deformation within the design tolerance.

Figure 2.2 Four phases of fixture planning



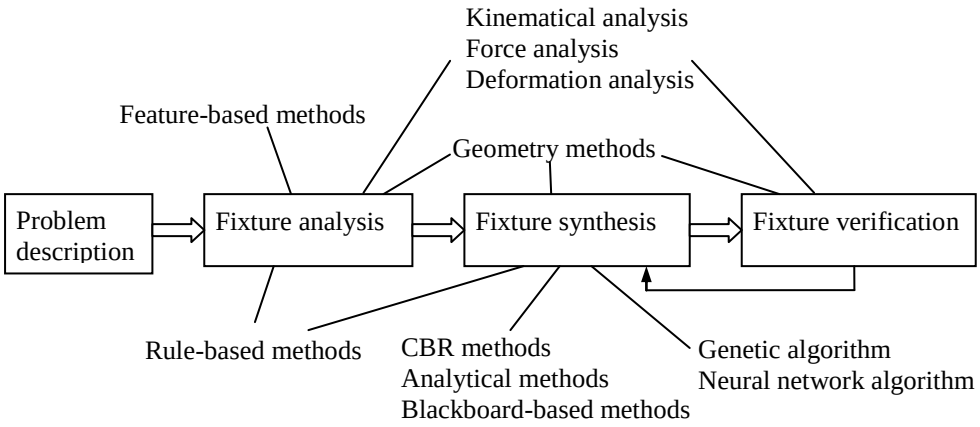
CAFP can be divided into four phases: problem description, fixture analysis, fixture synthesis, and fixture verification (Bi and Zhang, 2001), as shown in Figure 2.2. The problem description defines fixture design variables, design constraints and design objectives. In the fixture analysis, a workpiece-fixture interaction model is built in terms of geometry, kinematics, force, and deformation. The results of the analysis are then used to select the locating, supporting and clamping surfaces and points on the workpiece. The fixture

synthesis determines details of the fixture configuration including selection of fixture elements, placement of the elements in suitable locations, and generating fixture assembly plan. The fixture configuration is verified in respect to geometrical interference, locating determinant, clamp stability and machining tolerance in the fixture verification phase. If the design objectives are not satisfied, the result will be sent back to the fixture analysis phase for further improvements.

CAFP aims at the automation of fixture layout generation, fixture verification, and fixture assembly planning. The early work includes various interactive and semi-interactive fixture planning systems based on CAD systems (Hargrove and Kusiak, 1994, Hazen and Wright, 1990). Later research investigates the mechanism of fixture-workpiece interactions in respect to their geometry, kinematics, force, and deformation. Some optimization techniques were integrated into these methods. Trappey and Liu (1990) reviewed fixturing principles and automated fixture planning theory. Bi and Zhang (2001) presented a review of flexible fixture planning and automation. Cecil (2001) briefly reviewed current computer-aided fixture design approaches and indicated the trends of CAFD research as supporting cross-functional participation and concurrent engineering in a distributed and collaborative manufacturing environment.

Various approaches have been developed in CAFP to satisfy the four major constraints including geometrical constraints, accessibility constraints, force constraints, and deformation constraints. These approaches are classified in fixture analysis, fixture synthesis, and fixture verification as summarized in Figure 2.3.

Figure 2.3 Summary of CAFP approaches



In the fixture analysis phase, prominent methods have been developed including geometrical, kinematical, force, and deformation analysis methods, rule-based methods, and feature-based methods (Bi and Zhang, 2001, Pehlivan and Summers, 2008). The geometrical analysis checks the locating errors and interference among the fixture elements, workpiece and cutting tool path. The kinematical analysis ensures precise locating of a workpiece and accessibility of cutting tools and clamps. A sufficient clamp force is determined for the cutting conditions based on force analysis. The deformation analysis determines the potential deformation of a workpiece under the clamping and cutting force, which directly affects the final dimensional tolerance of the workpiece. Finite element analysis (FEA) is usually used for the deformation analysis. Feature-based methods and rule-based methods are applied for the part representation and fixture planning. Similarly, in fixture verification, the generated fixture configurations are evaluated using geometrical analysis, kinematical analysis, force analysis, and deformation analysis methods.

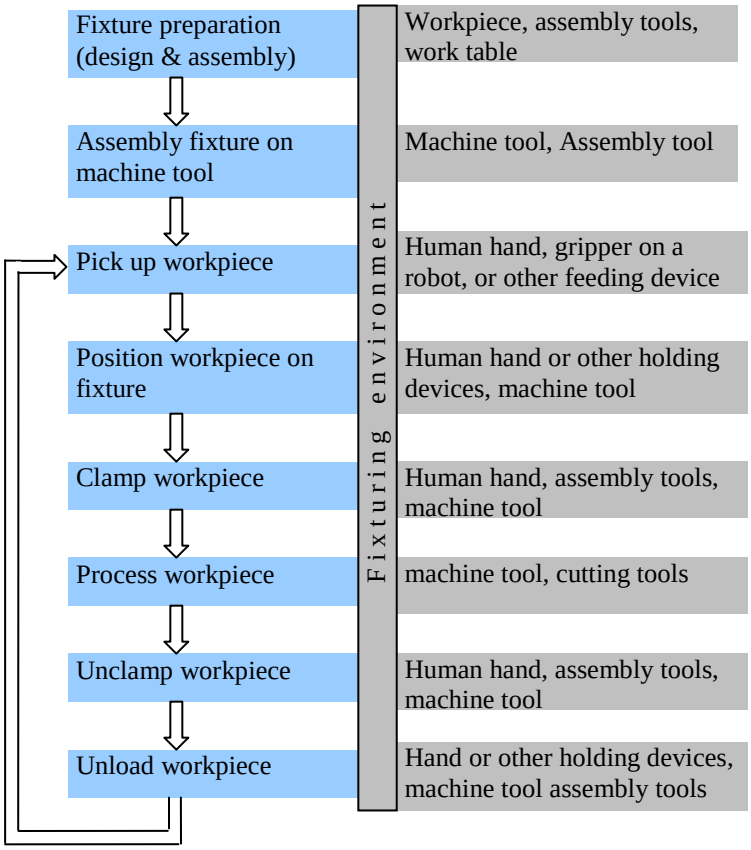
In the fixture synthesis phase, the locating, clamping, and supporting layout are determined. Analytical methods, rule-based methods, blackboard-based methods, geometry methods, and Case based reasoning (CBR) methods can be used for the fixture element selection and fixture layout planning. Optimization techniques such as genetic algorithm and neural network algorithm can be integrated into the workpiece-fixture mathematical model to search for a better fixture configuration. Rule-based expert systems are not theoretically generalized because the application of these systems in reality depends on the construction of a huge rule base to cover a sufficient domain (Trappey and Liu, 1990).

2.2 Related issues in fixture planning

The fixture assembly process can be divided into two main stages: assembling fixture itself and loading workpiece in the fixture. Figure 2.4 shows a workpiece handling process and its fixturing environment. Firstly, the fixture is designed, made and assembled at a separate place. Then it is assembled on the worktable of a machine tool. In some cases, it is also possible to assemble the fixture directly on the worktable of a machine tool. Then a workpiece is picked up from the storage area and positioned in the fixture. After the workpiece is clamped, it is machined according to the process plan, such as milling and drilling. Finally, the finished workpiece is unclamped and unloaded from the fixture. A new unprocessed workpiece is picked up. The process repeats until all workpieces in a batch have been processed. The fixturing environments in each step are illustrated in the right side of Figure 2.4. The related environmental factors include machine tools, assembly tools, human hands or grippers on a robot, and cutting tools. Currently, only a few

simulation-based fixture planning systems take some of these factors into account to verify the generated fixture plan. The fixture environmental factors were seldom considered in the early stage of fixture planning which may lead to fixture redesign or modification.

Figure 2.4 A workpiece handling process and its fixturing environment



2.2.1 Feasibility analysis of assembly tools

The use of proper tools in fixture assembly is one of the key factors to reduce production setup time. Tool feasibility affects the efficiency in fixture assembly and disassembly. A key tooling consideration for fixture assembly planning is to quickly determine the spatial

feasibility of tools in the design stage. Here tools refer to a wide variety of assembly tools including most hand tools and power tools which are required in fixture assembly. A proper tool is often selected from a set of tools, and then the tool is verified for its feasibility to perform a specific operation within the limit space. The verification is necessary for fixture elements to be assembled, disassembled, or for a part to be clamped in the fixture. The assembly tools are normally chosen based on personal experience and testing. Improper tool usage may lead to re-tooling, long lead time, and even design change of the fixture. Therefore, a fixture assembly plan should consider fixture elements and assembly tools simultaneously. Tool verification should be performed early in the fixture assembly planning to enable a feasible and complete fixture design.

Recent research efforts have addressed assembly tool feasibility problems in terms of tool representation and geometrical reasoning. However, some existing tool reasoning methods are either computationally expensive, or mainly based on virtual simulation. These approaches are not efficient in tool feasibility analysis for fixture assembly or disassembly process. On the other hand, the tool complexity and varieties raise difficulty for unskilled operators to select suitable tools. Less attention has been paid to reason the assembly tool applied in fixture assembly. Thus, there is a need for tool feasibility analysis in fixture assembly planning.

2.2.2 Fixture assembly planning

Fixture assembly planning generates the fixture assembly sequence, simulates the assembly process and evaluates the assembly plan. By analyzing the feasibility of a fixture, the

quality of a fixture design can be assessed and improved. The fixture assembly process can be optimised via adjusting the assembly sequence of fixture elements, reducing the number of assembly tools used, and easing the assembly. Eventually, the optimum fixture design is determined with a minimum life cycle cost.

Fixture assembly planning is similar to product assembly planning with some special characteristics. While the geometrical constraints are the same as product assembly, its assembly sequence is restrained by fixture functionality. The research in product assembly planning usually consists of three aspects: the representation schemes of a product for assembly, assembly sequence generation, and assembly sequence optimization based on all feasible assembly sequences. Good representation scheme of a fixture is the basis of fixture assembly planning. The scheme should be able to represent a complete and complex fixture. Many assembly and disassembly planning and optimization approaches have been developed based on these representation schemes, including operational research approaches (Yu and Li, 2005), Wave Propagation (Srinivasan and Gadh, 1998, Mascle and Balasoiu, 2003), genetic algorithm (Lazzerini and Marcelloni, 2000), ant colony algorithm (Wang et al., 2006), Petri Net (Moore et al., 2001), and motion planning approach (Sundaram et al., 2001). Nevertheless, fixture assembly planning technologies are not mature yet. There is a need for more generic and efficient assembly sequences planning approaches. In addition, most existing approaches require user interventions for fixture assembly planning and no fixture assembly sequence is generated. A fully automated approach is required to generate fixture assembly sequence with the consideration of its geometrical and functional constraints.

2.2.3 Fixture loading and unloading

One of the essential activities in manufacturing is the precise placement of a workpiece into a fixture, which is called fixture loading. Fixture loading is usually manipulated by a human operator or robotic manipulator. As mentioned before, after a fixture is assembled on the worktable of a machine tool, a batch of workpieces are repeatedly loaded into and unloaded from the fixture. Although much research has been conducted in fixture design and analysis, little attention was paid to fixture loading and unloading. Due to the geometrical diversity of workpieces and the complexity of the operating environment, it is required to conduct motion planning for fixture loading before actual operation. Successful motion planning in fixture loading is essential to save cost and reduce lead time for both human operators and robotic manipulators. It also helps to evaluate the fixture design alternatives in terms of ease of loading and unloading.

Given the start position and goal position of a workpiece, the optimal motion planning of fixture loading is to find a least-cost collision-free moving path of a workpiece from start position to the goal position. The cost of the path may be an arbitrary function such as the moving distance, the effort made to move, or the combination of several factors. The fixture loading can be analyzed by global accessibility and local accessibility. Global accessibility refers to workpiece's moving from start point to near fixture position. Local accessibility refers to the workpiece's moving from a near fixture position into a fixture. Much research has been conducted on local accessibility using kinematics theory. However, the global accessibility in fixture loading is required to find an optimal path in a

complex virtual environment. The fixturing environment may include many obstacles, such as machine tools, cutting tools, buffers, and fixture components.

2.3 Approaches to feasibility analysis of assembly tools

To assess an assembly tool's feasibility to assemble or disassemble a connector, assembly tools should be able to approach the applied component, and the tool's working space is interference free from other objects. Geometry based methods are mostly used for the feasibility analysis such as collision detection and global accessibility cone. Related research in areas of fixture verification, general assembly tools' analysis, CMM tool path planning are surveyed.

Accessibility analysis in fixture verification consists of two aspects, namely, the fixture elements, such as locators, can be approached by the fixturing surfaces of a workpiece when loading the workpiece, and the fixture elements are interference free during fixture assembly and various operations. Three types of interference may happen for a fixture: fixture element Vs. fixture element, fixture element Vs. workpiece, and fixture element Vs. cutting tool. Li et al. (1999) evaluated the accessibility of a fixture component to a workpiece's surface by testing the obstruction at grid points of the surface. The accessible direction is restricted to the normal direction of the surface. But the workpiece was separated from the fixturing environment. Kumar et al. (2000) developed an automatic fixture planning system. The interference between machining tools and fixture elements is detected using a cutter swept solid generated by tool motion silhouettes. The static interference test is performed using CAD built-in functions. Wu (2001) presented a systematic

geometric methodology for fixture planning. The motion feasibility in workpiece loading and unloading is analyzed based on the concept of visible cone and pivotal locator.

Most research on tool feasibility analysis in fixture assembly planning is to check the interference of cutting tool and fixture components. Hu and Rong (2000) performed the accessibility analysis of machining tool against fixture components for fixture design verification by expanding the 2D contour of fixture components and shrinking the tool as a moving dot/line segment. They used grid to exclude irrelevant fixture components for checking the tool path. Ilushin et al. (2005) presented a method for collision detection of multi-axis Numerical Control (NC)-machining tool based on space division techniques and polygon/surface-tool intersection algorithm.

Wilson (1998) incorporated the assembly tool reasoning into the assembly planning system with a detailed tool description. An assembly tool is represented as a use volume encoding the minimum free space in an assembly tool application, and placement constraints are used to locate the use volume in the assembly position. However, his approach is computationally expensive. The interference checking is performed by computing tool use volume's configuration space (C-space). Gupta et al. (1998) combined a collision detection method with randomized via-point path planners to avoid the expensive computation of the C-space. The articulated tool representation is used to represent assembly tools. Both of them assume that tool use volume and placement constraints are not variable during a tool application and only 1-DOF assembly tools were discussed in their research. Lazzerini and Marcelloni (2000) realized the importance of assembly tools and tool changes in optimal assembly sequence planning. However, the tool analysis has

not been incorporated into their work. Some commercial CAM software, such as VERICUT 6.1.2, supports a full tool path collision detection including cutting tools with fixtures and clamps, cutting tools with the part, and the tool shank and holder with the part (CGTech Ltd., 2007).

In many other application areas such as tool path planning for coordinate measuring machine (CMM), tool feasibility analysis has been well studied leading to several useful algorithms. Spyridi and Requicha (1990) proposed global accessibility cones (GACs) to represent feasible directions for a probe of CMM when measuring a point on an object. Their algorithm was primarily based on Gaussian image and Minkowski operations. Later Spitz et al. (1999, 2000) extended the algorithm from exact computation to discrete approximation of GACs by using computer graphics hardware to achieve a high processing speed. Various approaches to efficiently generate GACs at a single point were proposed by many researchers, such as ray casting technique and Boolean operations (Jackman and Park, 1998, Lim and Menq, 1994, Limaiem and ElMaraghy, 1997). Chung and Peng (2006) presented a new approach for tool reasoning in assembly and disassembly planning to reduce the computational cost for assembly tool reasoning. They constructed a discrete global accessibility cone with depth (GAC^d) to represent the obstacles around the fastener where the depth value is used for evaluating the tool feasibility. However, they failed to exclude the irrelevant objects from the calculation which led to inefficiency for a complex virtual environment. Moreover, tool reasoning was based on separated tool-application position instead of tool use volume which caused extra computations for in-between tool position. Finally, their tool representation was not complete due to the ab-

sence of tool end representation. Table 2.1 summarizes approaches and limitations related to feasibility analysis of assembly tools.

Table 2.1 Approaches to feasibility analysis of assembly tools

Resources	Application	Methods	Limitations
Li, 1999	Interference Workpiece and fixture compoents	Obstruction in grid points in surface	Restrict to normal direction, separate from fixture environment
Kumar, 2000	Interference of fixture and cutting tools	Cutter swept solid	Static interference checking
Wu, 2001	Motion feasibility in loading and unloading	Visible cone and pivotal locator	
Hu&Rong, 2000	Interference check of fixture and machinig tool	Shrink tools as a dot/line	Only 2D contour check
Illushin, 2005	Multi-axis NC tools and fixture/workpiece	Collision detection, space division technique	
Wilson. 1998	Assembly tool reasoning	Tool use volume' C-space	1-DOF tool reasoning, computational expensive
Gupta, 1998	Assembly tool reasoning	Collision detection, avoid C-space, use articulated tool representation	1-DOF tool reasonging
CGTech, 2007	Cutting tool path with fixture/part/shank/holder	Collision detection	
Spyridi,1990	CMM tool path planning	GACs with exact computation	Computational expensive
Spyridi, 2000	CMM tool path planning	GACs with discrete approximation	
Chung, 2006	Assembly tool reasoning	Discrete GAC[d], aritculated tool representation	Separate tool position reasoning, inefficient, incomplete tool representation

2.4 Approaches to fixture assembly planning

Fixture assembly planning is quite similar to product assembly planning. In product assembly planning, various representation schemes for assembly planning have been developed including graph-based representation, matrix-based representation, Petri Net-based presentation, and geometry-based representation. While some methods require users to interactively input the assembly relationships between components, the geometry-based method deduces the assembly relationships based on the geometry of the assemblies, which is advantageous due to the easy software development (Dong and Arndt, 2003). Among them, the matrix-based method is most promising for complex products. Other representation forms may be converted into matrices (Yu and Li, 2006). Chung and Peng (2006-1) developed a system for the product assembly and disassembly planning. The system can generate an efficient sequence of the product assembly and disassembly using an evolutionary algorithm. Peng and Chung (2007) conducted part accessibility analysis for product disassembly with the notion of using a digitized assembly directionality chart to represent approximate surroundings around a part. The analysis is mainly based on collision detection approaches in which the triangle patch-based collision tests are computational expensive. Boothroyd et al. (1994) did a lot of experiments to investigate the effect of obstructed access and restricted vision to the insertion of fasteners in general part assembly. The assembly time and cost are estimated to evaluate the quality of product design.

Several methods have been developed for fixture assembly planning, such as feature-based methods, fastener-based methods, and geometry-based methods. Bai and Rong

(1995) developed a method to represent modular fixture elements. A modular fixture element assembly relationship database (MFEARDB) was established based on assembly features and mating relationships. Yi and Nekey (1996) described a fastener-based approach in fixture assembly planning. Part Liaison Graph (PLG) and Fixture Liaison Graph (FLG) were generated to represent the assembly relationships. Dai et al. (1997) presented a modular fixture design and assembly system based on geometric analysis. A modular fixture tower database (MFTD) was generated to represent the assembly relationship of fixture elements. The fixture towers were selected and placed in the assembly. Ma et al. (1998) presented a computer-aided fixture design system named FIX-DES. Feature-based representation was used to describe the contact surface and the functionality of fixture elements. The assembly relationships between two fixture elements were automatically generated and rules were derived for the reasoning. The output of the system was a list of fixture elements and 3D drawings of fixture design. Kakish et al. (2000) proposed a knowledge-based universal modular jigs and fixtures design system (UMJFS). Joneja and Chang (1999) integrated the fixture planning into setup planning in which a generalized geometric and functional representation scheme for fixture elements was developed. The fixture used in different setups are generated and assembled automatically, based on certain rules.

Other research on fixture assembly planning is based on simulation in VEs through user intervention. Peng and Liu (2006) developed a VR-based modular fixture design system in which the user is allowed to manipulate the virtual models in a virtual environment for the fixture design and assembly. Rajan et al. (1999) presented a VR-based motion track-

ing system to evaluate product assembly sequence and jig design. The ergonomic analysis is performed by evaluating the recorded user hand's motion in the assembly. However, this method needs human intervention to assemble the product which takes a lot of time and effort. Peng (2006) developed a VR-based modular fixture design system. The user could move and position fixture elements for fixture design and assembly in a virtual environment. But the fixture assembly planning system was not implemented in the Web-based environment and no fixture assembly plan was generated. The assembly tool feasibility was not mentioned in their research. Wang et al. (2007) developed a product semantic modeling approach to interactively generate, evaluate, and simulate the assembly plan. Li (2008) presented an interactive VR system called VFDAS for fixture design and assembly that includes fixture element selection, fixture layout design, assembly and analysis. The system can simulate various physical behaviours for virtual fixture elements according to Newtonian physical laws, including gravity, friction, collision detection, mass, applied force, reaction force and elasticity. Peng et al. (2009) developed a desktop VR system called MF-VADS for modular fixture configuration design and assembly. A user can pick up fixture components and assemble them together in a VE. The collision detection method is used to avoid object overlap during assembling a fixture. However, the system lacks an automated evaluation module to generate assembly sequence plan.

2.5 Approaches to fixture loading and unloading

A fixture's reliable loading and unloading is an important issue in fixture evaluation. Fixture design alternatives can be assessed based on loading feasibility or ease of loading.

Kinematic analysis is widely used for the planning of fixture loading and unloading. Asada and By (1985) generated a kinematic model in the analysis of deterministic locating, accessibility and detachability, bilateral restraining and total restraint. The accessibility and detachability in workpiece loading and unloading are represented as the existence of an admissible small displacement from the workpiece's desired final location. Kang et al. (2003) established a geometric and kinematical model for fixture verification. The relationship of external forces, workpiece displacement and fixture deformation is formulated and the locator layout is optimized. Schimmels (1992) presented a method for loading a given fixture using generalized damper compliant motions, and concluded that a robust loading strategy exists for all deterministic fixture designs in the absence of friction. Lu (2000) described a motion planning approach to generate the folding sequences for a carton. The fixture used to folding cartons is kinematically modeled as a robot manipulator with joints and branching links. Yu (1998) presented an algorithm for generating sensor-based motion plans in 3-2-1 fixture loading. The fixture loading is divided as two phases: translational compliant motions, and combined compliant motions of both rotation and translation. However, these approaches are limited to local motion planning to move a small displacement from the workpiece's near fixture position into a fixture. The global motion planning considering all the obstacles in a complex environment has not been addressed. Jang et al. (2005) used a stereo camera mounted on a robot to capture the workspace when loading a part. The object accessibility is tested by utilizing visibility query based on real-time robotics manipulation**.** Table 2.2 summarizes the approaches to fixture loading and unloading. As potential methods in fixture loading and unloading, motion planning and optimization algorithms are surveyed in the following subsections.

Table 2.2 Approaches to fixture loading and unloading

Resources	Application	Methods	Limitations
Asada & By, 1985	Workpiece loading/unloading	Kinematic analysis, a small displacement exists	Local accessibility /detachability
Kang, 2003	Workpiece displacement	Geometrical & kinematic analysis	Local accessibility
Schimmels,1992	Fixture loading	Damper compliant motions	
Lu, 2000	Carton folding sequence	Kinematic model, motion planning method, fixture modeled as robot	Local accessibility
Yu, 1998	3-2-1 fixture loading	Sensor-based motion planning with translational /rotational compliant motions	Local accessibility
Jang, 2005	Loading part	Visibility query, stereo camera capture workspace	Global accessibility

2.5.1 Motion planning algorithm

Motion planning is to find a collision-free geometrical path in a virtual environment with obstacles. Motion planning algorithms are widely used in robotics, artificial intelligence, control theory, and computer graphics to convert high level specifications of tasks from human to low level description of motions (Lavalle 2006). The difficulty of motion planning greatly depends on the complexity of geometries and obstacles, the degrees of freedom (DOFs) of the object, and the objectives of the planning. In manufacturing, motion planning aids product assembly, disassembly, and maintenance to avoid costly construction of physical models. The motions of human or robotic operators can be planned to save enormous time and expense. For some robots with a large number of DOFs, the trajectory path may be required to combine both path and posture of a robot. In addition, a

complex virtual environment contains tens and thousands of obstacles. Thus a motion planning algorithm needs to be specific to the problem for an optimal and fast solution.

Motion planning has become an active research area since the notion of configuration space (C-space) was presented by T. Lozano-Perez in the late 1970s. In C-space, the robot is shrunk as a point while the obstacles are expanded to facilitate the path searching. During the 1980s, researchers in algorithmic motion planning proposed exact algorithms which pursued an exact representation of the obstacles in C-space and then used graph search techniques to search for a path. The exact algorithms are theoretically complete, but only a few of them are efficient and easy to implement. Many of them are difficult to understand and implement leading to impractical applications (Lavalle 2006). These algorithms include roadmap methods (Canny 1998), exact cell decomposition methods (Schwartz 1983), star-shaped roadmap representation for low-DOF robots (Varadhan 2005), to name a few. Later, practical approximated algorithms were proposed to avoid complex and inefficient implementation of exact algorithms. Approximated algorithms use implicit obstacle representations, such as cells or grids to conduct a random search. These algorithms are resolution complete but difficult for high dimensional spaces due to the numerous cells or grids required.

Sampling-based motion planning algorithms are more approximated but practical approaches. These approaches use a countable number of samples to represent the infinite state space. These samples are checked for interference with obstacles by collision detection modules to avoid the explicit construction of the obstacle region. Roadmaps are built randomly to represent collision-free paths in 3D space. Sampling-based algorithms are

probabilistically complete and sufficient for high dimensional spaces. RRT and PRM are two particular approaches. Kallmann presented a motion planning algorithm based on probabilistic roadmaps for the animation of human-like characters with 22 degrees of freedom (Kallmann 2003). Some hybrid algorithms combine two or more motion planning techniques for performance improvement. Zhang presented a hybrid motion planning method that combines approximated cell decomposition with probabilistic roadmaps. Their hybrid algorithm gains 10 times efficiency improvements over complete motion planning algorithms (Zhang 2007). Other methods, such as vision-based algorithms, can also be used for path planning. This research only considers the geometry-based path planning as the input data nature and problem definitions.

2.5.2 Optimization algorithms

After the C-space was represented as a graph, searching for an optimal motion plan is conducted by optimization algorithms. The searching approaches include breadth first search, depth first search, and best-first search. Breadth first search visits all possible vertices in each step to proceed to next step, such as Dijkstar's algorithm. Depth first search extends to the deepest vertices along one direction. Best-first search chooses the best solution in each step. The A-star search algorithm extends Dijkstar's algorithm by estimating the minimum cost at each step to select the best node to expand (Dechter 1985). Lozano-Perez used A* search technique to find the optimal path for robot manipulators to avoid collision of a robot's links with obstacles (Lozano-Perez 1997). The C-space for a manipulator with n degrees of freedom is represented by a tree of slice projections. The process recursively calculates the possible rotating range of a link. For each discrete posi-

tion of the previous link, the moving range of the next link is calculated. The resulted region graph serves as a basis of the path search. Ariadne's clew algorithm consists of two parts: exploring and searching algorithms (Mazer et al., 1998). Both are expressed as optimization problems and solved by Genetic Algorithm. Exploring algorithm represents the accessible space by placing landmarks uniformly in the C-space. To do this, it spreads the landmarks as far as possible in the C-space by maximizing the distances between them. Searching algorithm checks if the current landmark leads a path to the goal. Points colliding with obstacles are transformed backwards to avoid collision.

2.6 Implementation environments for fixture planning

The CAFP systems can be implemented as CAD-based systems, VR-based systems, and Web-based systems. Although the majority of CAD systems are capable of describing the geometrical features of mechanical components by points, curves, surfaces, and primitive solids, most of CAD systems currently lack an advanced built-in functionality to generate an assembly plan and to evaluate the feasibility of the assembly plan during the design stage. In addition, CAD systems have some drawbacks in collaboration and visualization. Firstly, it is not easy to emigrate a CAFP system from one CAD software to another because different CAD software systems use different data structures. Secondly, it is troublesome for the inter-enterprise collaboration and information sharing because a CAFP system often relies on a specific operating system. Fixture planning service providers have to spend a lot of time and efforts to solve the information exchanging problems. Finally, CAD systems only provide limited visualization which may lead to accessibility

and ergonomic related problems on the physical prototype. VR-based systems provide a better 3D fixture viewing than CAD-based systems. As a result, Web-based fixture assembly planning tools are required for the use of existing CAD models. Web-based systems for fixture planning enable information transferring, collaboration, and the better visualization.

The early CAD-based fixture planning systems provide an environment for modeling, visualization, positioning, and modification of a fixture. Fixture elements were stored in a database. Many functions have been realized in fixture planning systems including automatic fixture layout planning, fixture elements selection, fixture configuration verification, and assembly sequence planning. Based on the degree of design automation, these systems can be classified as the interactive (Fuh et al., 1995), semi-automated (Dai et al., 1997), and automated fixture design systems (Hou and Trappey, 2001).

VR-based fixture planning enables fixture visualization and manipulation in an intuitive way which allows direct interaction with fixture models in VEs, such as positioning fixture components using virtual hands. Li (2008) and Peng et al. (2009) are VR-based fixture planning systems by simulating assembly in VE.

Web-based fixture planning systems have emerged in recent years. These systems provide 3D graphical interactions and information sharing to help fixture designers cooperating with other engineers in a distributed manufacturing environment. Research on the Web-based fixture planning is relatively new. Most existing systems provide limited

functions. The following sections review the Web-based fixture planning systems, the technologies used, and data integration of CAD models with Web-based system.

2.6.1 Web-based fixture planning systems

Web-based fixture planning systems require 3D model visualization and manipulation for the ease of design. Several approaches have been proposed in dealing with model representation in distributed manufacturing environments. Three-tier client-server architecture is generally used with a graphics API. Standard file formats such as STEP, WRL or polygonized representation of 3D models are stored on the server side, and the model can be retrieved and viewed on the client side. In addition, an important issue in the Web-based fixture planning is the information integration of fixture planning with other product design and manufacturing systems.

Wagner et al. (1997) implemented a "FixtureNet" fixture planning system over the Internet based on Brost-Goldberg's algorithm (Brost and Goldberg, 1994). Mervyn et al. (2003) presented an Internet-enabled fixture planning system in which the part and fixture elements are polygonized. The facet data of polygonized models are embedded in the XML file and stored on the server side. The system allows users to interactively select fixturing surfaces and fixture elements on the client side. Heuristic rules are incorporated into the fixture planning process. Pehlivan and Summers (2008) conducted a literature survey for information representation requirements in CAFP. Mervyn et al. (2006) developed CAFD information models for information exchange to support the integration of product design and manufacturing. The model information was stored in XML file.

Hunter et al. (2005) presented a knowledge model for fixture planning which consisted of two stages: knowledge representation and inference process description. The first stage extracted the knowledge of an object, such as the part geometry, machining process, functional and detailed fixture planning and fixture resources. The second stage described the fixture planning and interpretation rules. The information was modeled using Unified Modeling Language (UML) and coded in C++. A rule-based method was used. Kang et al. (2007) presented a hybrid CBR/KBR (Knowledge-Based Reasoning) system. New designs based on KBR were compared and adapted to existing fixture planning cases to get a satisfied fixture plan. XML was used as the file format for information and knowledge transfer over the Internet. Liu et al. (2007) presented an agent-based CAPP/CAFP integration system, in which XML is used as a file format for information and knowledge exchange between agents. Fan and Kumar (2005) presented a CBR fixture planning system in which a workpiece was represented as a set of features. The Extensible Markup Language (XML) file format was used for the case representation. Three-tier client-server architecture was implemented using Java as the programming language, and Java3D as the graphics API.

2.6.2 Technologies used in Web-based systems

A Web-based VE can be implemented using several techniques. The widely adopted architecture is a three-tier client/server architecture which consists of a Web server, an application server, and clients. The client side user interface can use HTML and Java applets, in addition to ActiveX and VRML client plug-in embedded in the HTML file (Roy and Kodkani, 1999). Java3D (Wang et al. 2002) and X3D (eXtensible 3D) (Gelautz et al.,

2004) are also utilized to simulate and manipulate 3D geometry over the Web. VRML is a standard language for describing 3D scenes on the Web. VRML documents can be viewed through a Web browser with an appropriate plug-in. It allows external programs to interact with objects in the 3D scene by using External Authoring Interface (EAI). X3D (ISO/IEC 19775:2004) is the successor of VRML97 (ISO/IEC 14772:1997) (Web3D Consortium, 2008). Similar as VRML EAI, X3D defines the scene access interface (SAI) to interact with the X3D world either within the world or from external programs. However, VRML and X3D have limitations in linking with database and acquisition of the real-time information of the scene objects. On the server side, JavaServer Pages (JSP), Java servlets, and eXtensible Markup Language (XML) can be implemented to provide various functionalities such as database management and rule-based reasoning. Conflict resolutions, security issues and reducing network traffic remain concerns in the implementation of Web-based collaboration systems (Wang et al. 2002).

2.6.3 Data integration of CAD models with fixture planning

Since a fixture is usually designed in a CAD system, it is required to integrate CAD models with fixture assembly planning. One method is to directly employ collaborative CAD modeling tools to share product information. Li et al. (2007-1) presented a collaborative design platform which allows users to construct and modify the same CAD models from various sites with different CAD systems. But some analysis functionalities are not available in the CAD systems for a specific task. The other is to transfer and visualize 3D models via Internet. Several neutral data formats have been proposed to share CAD models among different systems. A difficulty in generating assembly plan from CAD models

is that much information of the product assembly is not directly available from these CAD data formats. Some assembly constraints are lost or not recorded. Some data formats need the specific interface for explanation.

A number of neutral data formats are in use to share geometry information, such as IGES, DXF, SAT, STL and STEP. They are designed to facilitate the system-to-system data sharing between CAD/CAM systems. The international standards for product data exchange, STEP (STandard for the Exchange of Product model data), enables data sharing between different CAD/CAM systems by integrating product's life cycle data. Pan et al. (2003, 2005) presented a method to extract geometrical data from STEP files. The geometrical data are converted from the CAD STEP files for assembly analysis. The STEP file is first translated into XML via STEP-XML translator, XML file is then parsed using Java DOM. In particular, the extracted information includes faces, edges, vertices and coordinates of vertices of each part. The geometrical data are used to determine the interference relationships of parts in the assembly planning. Jayaram et al. (2004) used a hybrid method to generate the polygonal representations of CAD assembly models and to extract original assembly constraints for the use of assembly planning. Cicek and Gulesin (2007) presented a method to recognize CAD models by analyzing their face adjacency relations and attributes in STEP files. Zha and Du (2002) developed a product data modeling method for the integration of design and assembly planning based on STEP files. However, the graphical visualization of geometry models and assembly planning over the Internet remains a challenge for the STEP-based assembly planning as this format is not intended to be used in Web-based environments. For STEP-based conversion, it is not

easy to access the geometrical data in STEP files because a specific development environment is required using the standard data access interface (SDAI). It is either costly by using commercial software or time-consuming by developing own software to convert STEP files. Moreover, it is difficult to visualize and manipulate the STEP-based geometry over the Internet. Commercial software called ST-Developer has been developed by STEP Tools Inc. Its ST-Viewer can read STEP files and convert them to the Open Inventor file format. Only product images generated by ST-Viewer can be translated into the VRML format and shown on World Wide Web (STEP Tools, Inc. 2008).

VRML is a widely used standard data format for representing 3D triangle mesh models over the Internet (Li et al., 2007-2). Compared with the system-to-system transferring file formats, VRML is especially suitable for applications in collaborative product development environments. VRML is easy to use and has a good mechanism for Web-based applications. Most 3D CAD models are generated as boundary representation models. 3D CAD files are usually huge in size. In order to transfer and visualize the CAD models over a networked system, they are usually discretized into triangle meshes for visualization. VRML is an open standard maintained by ISO/IEC to interactively encode, animate 3D objects and scenes over the Internet, Intranet and local client systems. VRML documents can be viewed through a Web browser with an appropriate plug-in. It allows external programs to interact with objects in 3D scenes by using EAI. VRML has been used in a variety of application areas for visualization and animation (Web3D Consortium, 2008). VRML file is small in size and suitable for transferring over the Internet. By extracting only useful information from VRML files, the file size can be further reduced.

VRML is fully supported by most current CAD systems. X3D (Extensible 3D) is the enhanced successor of VRML, which is integrated with XML (Extensible Markup Language). X3D provides more features than VRML97 (Web3D Consortium, 2008). However, X3D file format is not directly available in most CAD systems. It would generally require multiple steps of conversion through several formats to transfer CAD models to X3D format. Nevertheless, X3D is the trend as long as new releases of CAD systems support X3D file format.

A Web-based assembly planning approach has been presented over the Internet which uses VRML representation for visualization (Chung and Peng, 2006-2). Liu et al. (2008) extracted the assembly tree from CAD models in a CAD platform. The CAD models are stored as VRML files for visualization over the Web. The assembly tree and model information is integrated together for assembly planning and simulation. Hren et al. (2006) use VRML as data viewing and sharing tools over the Web with limited assembly manipulations.

2.7 Construction and performance improvement for large VEs

Large VEs have become an effective tool in engineering design of complex products or construction of large buildings. A large-scale VE normally consists of numerous geometrical models, and results in large computation loads for the VR engine to handle in real-time. In addition, the computer memory required for a complex VE may become very

demanding. Although the computer capability has been increased significantly, a large-scale VE can severely influence the performance of a VE system.

Model management techniques are frequently used for a VR engine in rendering complex VE to maintain the simulation quality. There are several model management approaches, such as level of detail (LOD), cell segmentation, off-line pre-computations, and database management. A combination of these techniques is often used to get the improved simulation. Cell segmentation methods partition a virtual world into smaller parts, or cells. The model complexity can be significantly reduced by rendering only a small amount of objects within the current visible world. As the viewpoint moves, new cells are dynamically added and removed from the scene. Thus, the set of polygons that appears in each view can be kept small (Pimentel and Teixeira, 1993). Cell segmentation can not only improve the performance of a simulation, but also benefit designers. One important role of simulation is to facilitate the modification during the design stage. For example, an architect may change a door position that is located in a room or hallway. In this case, it is possible to only modify this small region, without changing other parts of the whole model.

Airey et al. (1990) presented an automatic model-space subdivision and potentially visible set calculation methods. Potential visible set (PVS) is the union of visible polygons for all viewpoints in a cell. For any viewpoint in a cell, rendering the PVS for that cell gives the same scene as if the whole model is rendered. Since the PVS is much smaller than the total size of the model, rendering takes much less time. The recursive model-space subdivision identifies and numbers planes according to a partition priority using bi-

nary space partitioning. Where occlusion refers to how well a component can hide other polygons in a sub-assembly, balance refers to how evenly a plane separates the model, and split refers to how little the plane splits individual polygons. When the cell has holes in its boundary, for example, a room has doors, certain external polygons have to be added to its PVS. This problem is called the volume to polygon visibility problem, and the holes in the cell boundary are called portals.

Forming potentially visible sets (PVS) based on the attribute of cells is the next concern after the cell segmentation, which also refers to the visibility calculation. The main algorithms can be classified as point visibility culling and region visibility culling. The point visibility culling method computes the visibility from a point which is applied in each frame during rendering (Greene et al. 1993, Coorg and Teller, 1996, Zhang et al. 1997, Bittner 1998). The region visibility culling method computes visibility for a region rather than a single point (Cohen-Or et al. 1998, Leyvand 2003, Saona-Vazquez 1999, Wonka 2000). Since it is inherent in a 4-dimensional problem, it is more difficult for the region method to use than the point method.

A visibility culling algorithm has been used to show a large model. Visibility culling algorithms including exact pre-processing algorithms (Teller and Seaquin1991), and conservative runtime algorithms (Roden and Parberry, 2005) have been developed. The visibility pre-processing method was presented for axis-aligned or axial architectural models. The whole model is subdivided into cells along opaque surfaces. An adjacency graph of the cells is formed along the non-opaque portals. The cell-to-cell visibility is then computed for each cell according to sightlines existence between pairs of cells. When ob-

server moves in a real-time simulation, the relevant PVS are retrieved from storage. An approach for dynamically determining PVS of cells in real-time simulation was developed by Luebke and Georges (1995). It uses a screen space projection to compute a conservative estimate of PVS at rendering time, providing increased interactive performance for large architectural models. Another technique that further simplifies the rendering process is by replacing the cells visibility through a portal with the texture (Aliaga and Lastra, 1997).

The region visibility, where the computed visibility is valid for a region rather than a single point, is considered more difficult to implement than the point visibility. This kind of problem draws more attention than the point visibility recently. The methods of the region visibility take advantages of time and spatial coherence. The computational cost of the visibility calculation is distributed to consecutive frames. A conservative occlusion culling method was presented, which was based on factorizing the 4D visibility problem into horizontal and vertical components and then was solved asymmetrically. It divided the original scene into a kd-tree. During the algorithm execution, the kd-tree was traversed from top to down (Leyvand et al., 2003).

The previous research concentrates on the visibility algorithm. They normally tag each polygon with a number and store the data structure in a database. Instead of that, a practical solution is proposed in this research to form a local region and store it as a PVS group, which facilitate designers to modify the design in a small area. It is easy to implement for storing complex polygons relationship in real-time.

2.8 Limitations of the existing research

The operation feasibility of fixture planning is restricted by fixture environmental factors including machine tools, assembly tools, grasping devices, cutting tools, fixture components, and workpieces. Although many approaches to fixture planning have been developed, the neglect of environmental factors may lead to mismatch of the fixture analysis with actual fixturing feasibility in machining. Moreover, a relative "static" fixture configuration is generated and verified in the sense of not considering the possible motion of a workpiece in the workpiece loading and unloading, as well as the assembly trajectory of fixture components. A "dynamic" fixture planning can integrate fixture assembly motions with various environmental factors.

Research in fixture assembly planning has gained less attention than fixture design. Existing fixture assembly planning emphasizes on the representation of the assembly relationship of fixture elements. Most of these planning systems do not generate an assembly sequence plan. In addition, the fixture assembly has not been thoughtfully investigated in terms of global accessibility, motion planning, and working feasibility in complex manufacturing environments. Although local accessibility/detachability has been investigated in workpiece loading and unloading, there is a lack of global accessibility analysis methods for fixture-related operations in complex manufacturing environments. Motion planning in the fixture assembly is an important means to address the global accessibility problem.

There is also a need to facilitate the fixture planning and information sharing in VEs. Fixture planning has to be integrated into concurrent and distributed virtual manufacturing environments. Information sharing among geographically distributive teams of product design, fixture planning, manufacturing, and resource management is essential for a rapid response to the dynamically changed market. Web-based distributed fixture planning forms a platform for information sharing of part design, manufacturing, and fixturing.

Based on the above observation, this research will investigate approaches to dynamic fixture planning in virtual environments. Dynamic fixture planning will verify a fixture plan, validate the fixture assembly, and simulate motions in fixture applications in VEs. It enables intuitive model visualization, manipulation, and information sharing in the fixture design stage. The research includes fixture assembly planning, feasibility analysis of assembly tools, and motion planning for fixture loading and unloading with the fixturing environmental factors.

2.9 Chapter summary

This chapter reviews issues and various approaches to CAFP. The implementation environments for fixture planning are classified as CAD-based systems, VR-based systems, and Web-based systems. Limitations of current research and some promising research areas are identified in fixture assembly planning, feasibility analysis of assembly tools, motion planning in fixture loading and unloading in Web-based VR environments.

Chapter 3

Feasibility analysis of assembly tools

This chapter presents a new approach to tool feasibility analysis for fixture assembly planning. A tool workspace to operate the fixture is represented by a newly defined global accessibility cone with depth of a truncated half-line called $GAC^{d}_{(R)}$. An assembly tool is modeled as five articulated parts to fully describe the tool characteristics. The tool feasibility analysis verifies an assembly tool's feasibility applied on a fastener. In addition, both tool motion and tool placement constraints during the tool application are integrated into the tool geometric reasoning. The examples will demonstrate that the proposed approach is efficient and valid. The analysis result is validated by intuitive simulation of assembly tools applied on a fastener.

3.1 Assembly tools' modeling in fixture assembly planning

The fixture assembly planning determines the assembly sequence of fixture elements. After the fixture configuration design is accomplished, the fixture elements are made or se-

lected from a standard fixture database. For a modular fixture, the fixture elements, including a base plate, several locators, supports, and clamps are assembled on the worktable of a machine tool.

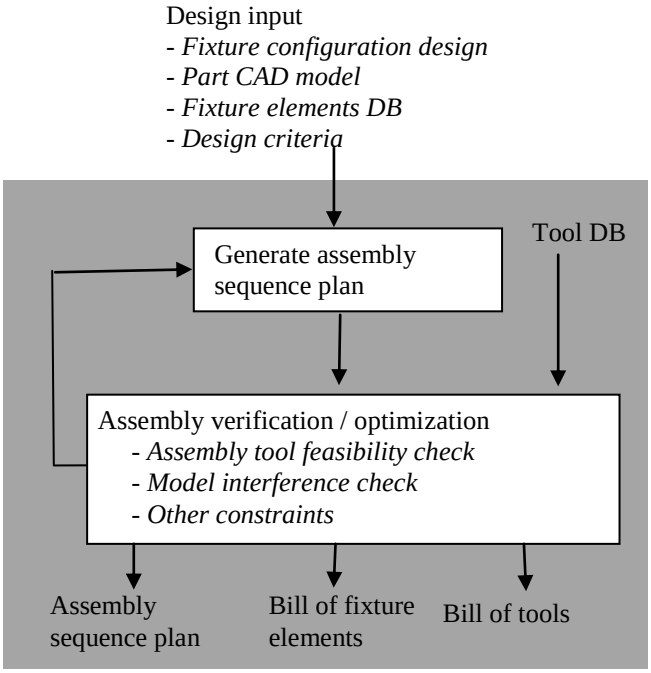
Tool feasibility analysis is critical to validate a fixture assembly plan and optimize the fixture assembly sequence. The operation constraints strongly influence the feasibility in implementing a fixture assembly plan. Improper fixture assembly sequences may cause difficulties in the tool application. Some tools may be very sensitive to the surroundings and one need to specify their relative positions to the fastener in the design stage. Moreover, proper tool assignments in the planning stage save operators' time and effort in determining and testing assembly tools. A fixture is subject to several constraints related to the fixed workpiece, such as functionality, tolerance, stability, accessibility, deformation, and structure requirements. Since complex fixtures consist of many fixture elements to fulfill these requirements, the tool motions around a fastener are restricted in a limited space during assembly operations.

3.1.1 The architecture of fixture assembly planning

The fixture design involves determining fixturing planes on the workpiece, selecting the fixture elements, and planning the assembly of these elements. Design constraints such as the process plan, operation schema and cutting forces are considered in the fixture configuration design. After the fixture configuration has been decided, a feasible assembly sequence plan is required to construct the fixture. The mating relationship between fixture elements can be described as an assembly tree structure (Dai et al., 1997). An assem-

bly sequence is planned based on the assembly relationship model. Figure 3.1 shows the architecture of the fixture assembly planning. The assembly sequence plan is generated based on fixture configuration, design variables and criteria. The plan is then verified in term of tool feasibility, geometrical interference, and other constraints such as fixturing positioning and deformation errors, and clamp stability due to the clamp force. If there is no feasible tool or there is any interference, the plan has to be modified. This process continues until a feasible assembly sequence plan is achieved. The output includes fixture element lists, tool lists, and a detailed assembly sequence plan.

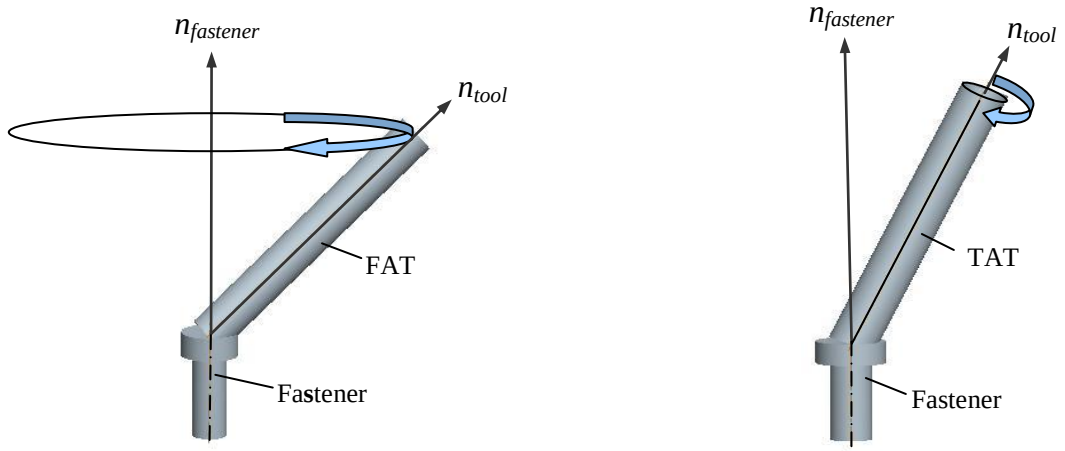
Figure 3.1 The architecture of fixture assembly planning



3.1.2 Assembly tool classifications

There are varieties of assembly tools used in fixture assembly and disassembly, such as hand tools and power tools. In this research, all the assembly tools are classified into two categories: fastener-axis action tools (FATs) and tool-axis action tools (TATs) (Chung and Peng, 2006-3). Figure 3.2 (a) shows the FATs which rotate about the fastener axis when rotating a fastener, such as various wrenches and ratchets with sockets. Figure 3.2 (b) shows the TATs which rotate about the tool axis such as screwdriver, power tools, and speeder with sockets. Besides the rotational movement, both FATs and TATs have a displacement along fastener axis with the fastener movement. A general model is defined to represent these two kinds of tools.

Figure 3.2 Two kinds of assembly tools: (a) FAT (b) TAT



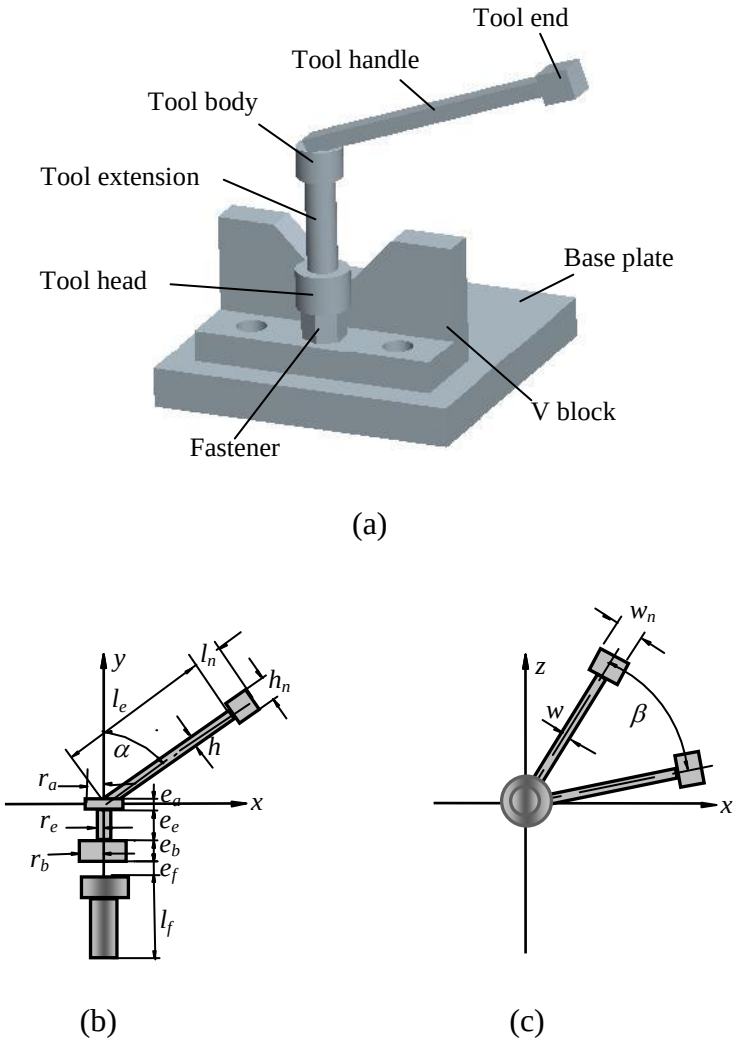
(a) (b)

3.1.3 Assembly tool modeling

An assembly tool can be modeled as five articulated parts: tool head, tool extension, tool body, tool handle and tool end as shown in Figure 3.3 (a). A tool head is the portion occupied by a fastener head. Right above the tool head is the tool extension. A tool handle is the portion between a tool body and a tool end, including the part grasped and applied by human hands or robot arms. The tool body links the tool extension and the tool handle together. The tool end is the portion beyond the tool handle. For instance, the battery part of a power wrench is defined as the tool end. In addition, a tool joint is defined as the connecting point of the tool body and the tool handle. The model can fully describe most of the assembly tools except some special tools.

Based on the tool dimension and its motions, seventeen parameters l_e, w, h, r_a, r_e, r_b, e_a, e_e, e_b, e_f, h_n, l_n, w_n, l_f, $\alpha_{\min}$, $\alpha_{\max}$ and β are used to model the assembly tool. The parameters are defined in the list of symbols. In particular, e_f is the relative distance from the top of the fastener to the bottom of the tool head which is used to adjust the tool position when applied on a fastener. In this research, l_f is assumed as the fastener's length, while in some cases the translational motion may be a minor value. The assembly tool modelling is illustrated in Figure 3.3. In Figure 3.3(a), a tool is applied on a fastener which connects two fixture elements together. The tool is abstracted as five parts. Figure 3.3(b) and 3.3(c) show the tool dimension parameters, as well as the tool movement parameters.

Figure 3.3 Assembly tool modeling (a) tool abstract model (b) xy-plane view (c) xz-plane view



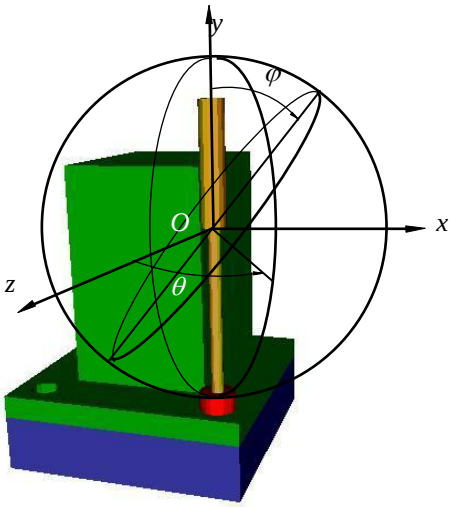
3.2 Global accessibility cone with depth of a truncated half-line $GAC^d_{(R)}$

A global accessibility cone with depth of a truncated half-line $GAC^d_{(R)}$, which is shown in Figure 3.4, is an extension of a GAC^d (Chung and Peng, 2006-3). GAC^d is a discrete unit sphere representing the surrounding obstacles information in a set of directions. The

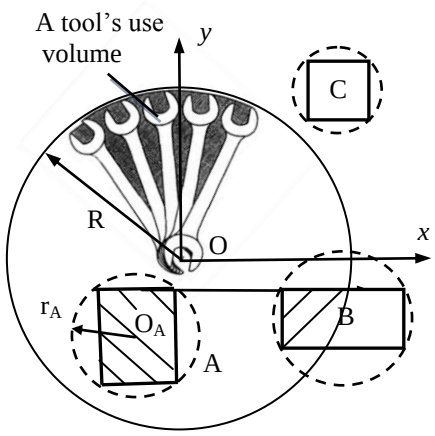
unit sphere is discretized into 180x360 pixels by 180 colatitude angles (φ) and 360 longitude angles (θ) in a spherical coordinate system. Unit vectors in a 3D space are formed from the centre point of a GAC^d to these pixels. Thus, 180x360 corresponding directions are formed and equally distributed on the unit sphere. $GAC^d_{(R)}$ excludes the obstacles which are farther from the sphere centre than a specified distance R. A truncated half-line (0, R) is a line segment of a half-line (Spitz and Requicha, 2000). R is the distance from the sphere centre to the endpoint of the half-line. As shown in Figure 3.4 (b), R is modeled as the maximum radius of tool occupied space (the tool's use volume) centered at the tool joint. It is clear that only the obstacles inside the R sphere may collide with the tool for a tool application. The $GAC^d_{(R)}$ can be displayed as a 2D accessibility map where a pixel in the 2D accessibility map represents a direction in a 3D space. Figure 3.4 illustrates the formation of $GAC^d_{(R)}$.

Figure 3.4 (a) shows the coordinate system of $GAC^d_{(R)}$. A screwdriver is applied on a fastener. The joint point O is defined as the center of $GAC^d_{(R)}$. In Figure 3.4 (b), the object A and the part of B (hatched section) are included in the formation of $GAC^d_{(R)}$, while the object C and outer part of B are excluded from the calculation. Figure 3.4 (c) is the 2D accessibility map of the $GAC^d_{(R)}$. The dark grey part indicates the non-accessible directions which are blocked by surrounding objects.

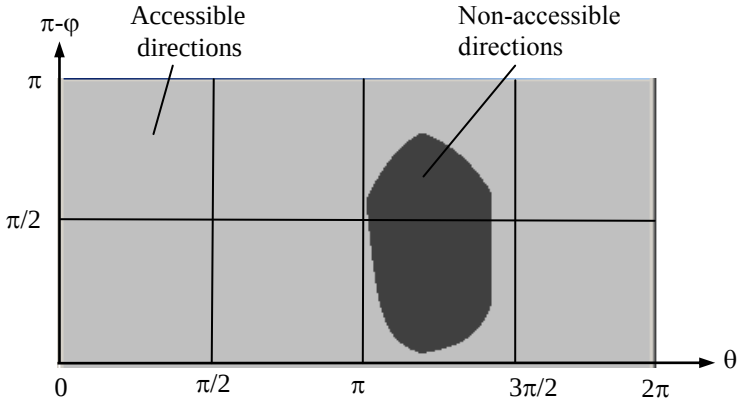
Figure 3.4 Formation of the $GAC^{d}_{(R)}$ (a) the coordinate system of $GAC^{d}_{(R)}$ centered at tool joint (b) object A and part of B which are inside R sphere are included in the construction of $GAC^{d}_{(R)}$ (c) 2D accessibility map of $GAC^{d}_{(R)}$



(a)



(b)



(c)

To form a $GAC^{d}_{(R)}$, the tool joint is set as the sphere centre O, which has a distance $H_j = 0.5e_a + e_e + e_b + e_f$ from the top of the fastener, and the fastener removal direction is

aligned to y-axis. Then each obstacle is evaluated to determine its distance from the $GAC^{d}_{(R)}$ center. For those obstacles closer than R, all triangle patches are included in the $GAC^{d}_{(R)}$ construction. For those obstacles partly inside the boundary, only triangle patches inside the boundary are included in the $GAC^{d}_{(R)}$ construction. As shown in Figure 3.4 (b), an enclosed sphere can be determined for an obstacle A. Let the sphere center be O_A and the sphere radius be r_A. If the distance $(OO_A + r_A) \leq R$, then obstacle A is inside the R sphere. This obstacle is included in the formation of $GAC^{d}_{(R)}$. Otherwise, obstacle A may locate outside the R sphere or cross the R sphere. In this case, if the distance $(OO_A - r_A) < R$, then part of the obstacle A may locate inside the R sphere. The distances from $GAC^{d}_{(R)}$ center O to each triangle patch in A is calculated. A triangle patch P_i is included in the formation of $GAC^{d}_{(R)}$ only if at least one point of a triangle patch is inside the R sphere. The pseudo-code is given in Figure 3.5.

Figure 3.5 Pseudo-code for computing $GAC^{d}_{(R)}$

```
For each obstacle A {
    Calculate sphere center O_A, sphere radius r_A
    Calculate distance OO_A
    if (OO_A + r_A) ≤ R { form GAC^d_(R) }
        else {
      if (OO_A - r_A) < R {
          for each triangle patch P_i
              if P_i is inside R sphere { form GAC^d_(R) }}}}
return the GAC^d_(R)
```

3.3 Feasibility analysis of assembly tools

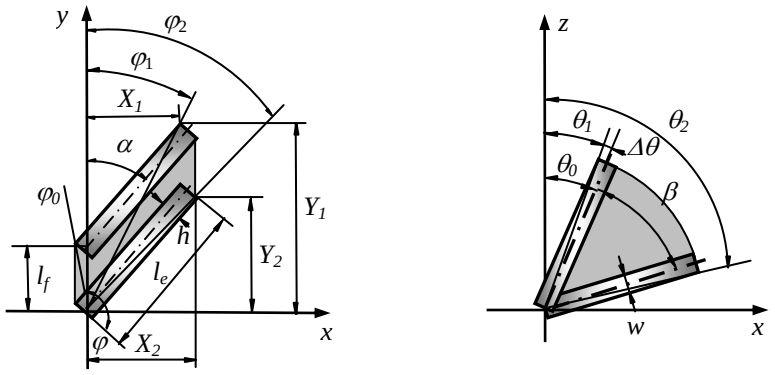
In this research, a tool' feasibility analysis is based on the tool's use volume. A tool's use volume is the minimum swept volume for the tool application (Wilson, 1998). If a tool's use volume can be placed in a collision-free position, this tool application is feasible. Instead of checking the tool's feasibility in every separate position during tool applications, the tool's use volume as a whole block is reasoned. Therefore, only boundary checking of the tool's use volume is required instead of checking the boundary in a tool's every position. The computation demand is reduced. The tool feasibility analysis is performed for the modeled five tool parts, respectively. In the following analysis, the feasibility of a tool application is represented as F, where F=1 or F=true represents a feasible tool application, and F=0 or F=false refers to an infeasible tool use.

3.3.1 Feasibility analysis of tool handle

The tool handle's use volume is formed by a translational movement along the fastener removal direction (*y*-axis), and the rotational movement about *y*-axis. Some tool handles can vary certain degrees in the access angle α while others can't. It is assumed that the initial position of the tool handle starts from a longitude angle θ_0, the searching process is executed by changing θ_0 from 0 to 2π to place the tool handle's use volume with minimum tool-application angle β. The tool access angle α may vary between the minimum tool access angle α_{min} and the maximum tool access angle α_{max}. The searching process for α starts from α_{min} to α_{max} at a longitude angle θ.

Assume the tool handle is placed at (θ_0,α), the feasibility analysis is performed at boundaries of the tool use volume along φ direction. Figure 3.6 shows the tool handle's use volume in the xy-plane and xz-plane. Four dividing angles φ_0, φ_1, φ_2, φ_3 along the φ direction and two angles θ_1, θ_2 along θ direction are defined in Equations (3.1) and (3.2).

Figure 3.6 Feasibility analysis of the tool handle



Equations (3.3) and (3.4) are derived to check the interference along φ direction when φ varies from φ_0 to φ_3 for a tool position θ, which varies from θ_1 to θ_2. $R(\theta, \varphi)$ is the depth at a pixel (θ, φ) of $GAC^{d}_{(R)}$. The distance $r_0(\varphi)$, $r_1(\varphi)$ are the line length at upper and lower boundaries shown in Figure 3.7. $\hat{\boldsymbol{T}}(\theta, \varphi)$ represents the unit vector on a pixel (θ,φ). L represents the projection of the maximum length of the tool handle in the tool's use volume on the xz plane. Similar analysis can be executed for the outer boundaries along θ direction when $\theta < \theta_1$ and $\theta > \theta_2$ which is omitted here.

$$\begin{aligned}
&\varphi_0 = -\tan^{-1}\left(\frac{0.5h\cos\alpha}{l_f + 0.5h\sin\alpha}\right)\\
&\varphi_1 = \tan^{-1}\left(\frac{X_1}{Y_1}\right) = \tan^{-1}\left(\frac{l_e\sin\alpha - 0.5h\cos\alpha}{l_e\cos\alpha + l_f + 0.5h\sin\alpha}\right)\\
&\varphi_2 = \tan^{-1}\left(\frac{X_2}{Y_2}\right) = \tan^{-1}\left(\frac{l_e\sin\alpha + 0.5h\cos\alpha}{l_e\cos\alpha - 0.5h\sin\alpha}\right)\\
&\varphi_3 = \pi/2 + \alpha
\end{aligned} \tag{3.1}$$

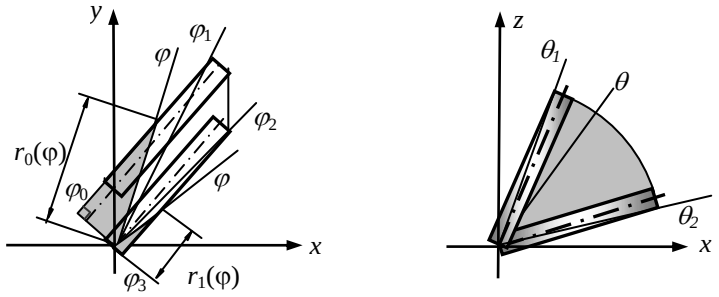
$$\begin{aligned}
&\Delta\theta = \tan^{-1}(0.5w/l_e)\\
&\theta_1 = \theta_0 - \Delta\theta\\
&\theta_2 = \theta_0 + \beta + \Delta\theta
\end{aligned} \tag{3.2}$$

$$F = \begin{cases}
1 \quad if\ [(\varphi_0 \le \varphi \le \varphi_1) \cap (R(\theta,\varphi) - r_0(\varphi) > 0)]\\
and \quad if\ \{(\varphi_1 \le \varphi \le \varphi_2) \cap [\left\| R(\theta,\varphi)\hat{\mathbf{I}}(\theta,\varphi)\cdot(\hat{\mathbf{i}} + \hat{\mathbf{k}})\right\| - L > 0]\}\\
and \quad if\ [(\varphi_2 \le \varphi \le \varphi_3) \cap (R(\theta,\varphi) - r_1(\varphi) > 0)]\\
0 \quad otherwise
\end{cases} \tag{3.3}$$

$$\theta_1 \le \theta \le \theta_2, \varphi_0 \le \varphi \le \varphi_3$$

$$\begin{aligned}
&L \cong (l_e^{\,2} + (0.5w)^2 + (0.5h)^2)^{1/2}\sin\varphi_2\\
&r_0(\varphi) = \frac{l_f\sin\alpha + 0.5h}{\sin(\alpha - \varphi)}, \varphi_0 \le \varphi \le \varphi_1\\
&r_1(\varphi) = \frac{0.5h}{\sin(\varphi - \alpha)}, \varphi_2 \le \varphi \le \varphi_3
\end{aligned} \tag{3.4}$$

Figure 3.7 Illustration of the tool handle's calculation



3.3.2 Feasibility analysis of the tool end

The tool end moves along the tool handle, therefore the feasibility analysis is similar to that discussed for the tool handle. Assume the tool handle is placed at (θ_0,α), the feasibility analysis of the tool end is executed at boundaries of the tool use volume along φ direction. Figure 3.8 shows the tool end's use volume in the xy-plane and xz-plane. Four dividing angles φ_0, φ_1, φ_2, φ_3 along φ direction and two angles θ_1, θ_2 along θ direction are defined in Equations (3.5) and (3.6). The feasibility analysis is executed based on Equations (3.7) and (3.8).

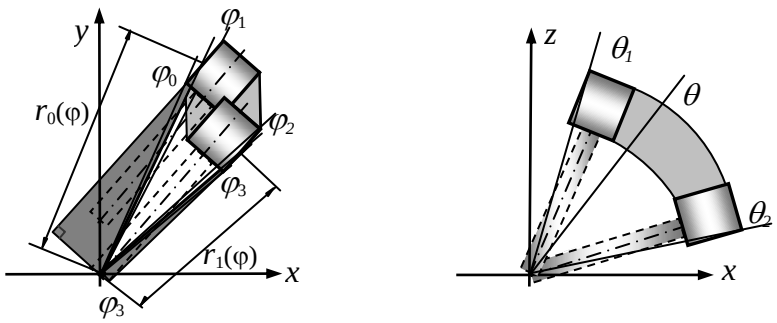
$$\begin{aligned}
\varphi_0 &= tan^{-1}\left(\frac{l_e\ sin\alpha - 0.5h_n\ cos\alpha}{l_f + l_e\ cos\alpha + 0.5h_n\ sin\alpha}\right)\\
\varphi_1 &= tan^{-1}\left(\frac{l_e sin\alpha - 0.5h_n\ cos\alpha + l_n\ sin\alpha}{l_f + l_e\ cos\alpha + 0.5h_n\ sin\alpha + l_n\ cos\alpha}\right)\\
\varphi_2 &= \alpha + tan^{-1}\left(\frac{0.5h_n}{l_e + h_n}\right)\\
\varphi_3 &= \alpha + tan^{-1}\left(\frac{0.5h_n}{l_e}\right)
\end{aligned} \tag{3.5}$$

$$\begin{aligned}
\Delta\theta &= tan^{-1}\left(\frac{0.5w_n}{l_e + l_n}\right)\\
\theta_1 &= \theta_0 - \Delta\theta\\
\theta_2 &= \theta_0 + \beta + \Delta\theta
\end{aligned} \tag{3.6}$$

$$\begin{aligned}
F = \begin{cases}
1 \quad if\ [(\varphi_0 \le \varphi \le \varphi_1) \cap (R(\theta,\varphi) - r_0(\varphi) > 0)]\\
and \quad if\ \{(\varphi_1 \le \varphi \le \varphi_2) \cap [\left\| R(\theta,\varphi)\hat{\mathbf{T}}(\theta,\varphi)\cdot(\hat{\mathbf{i}} + \hat{\mathbf{k}})\right\| - L > 0]\}\\
and \quad if\ [(\varphi_2 \le \varphi \le \varphi_3) \cap (R(\theta,\varphi) - r_1(\varphi) > 0)]\\
0 \quad otherwise
\end{cases}\\
\theta_1 \le \theta \le \theta_2, \varphi_0 \le \varphi \le \varphi_3
\end{aligned} \tag{3.7}$$

$$
\begin{aligned}
&L \cong ((l_e + l_n)^2 + (0.5w_n)^2 + (0.5h_n)^2)^{1/2} \sin\varphi_2 \\
&r_0(\varphi) = \frac{l_f \sin\alpha + 0.5h_n}{\sin(\alpha - \varphi)}, \varphi_0 \le \varphi \le \varphi_1 \\
&r_1(\varphi) = \frac{0.5h_n}{\sin(\varphi - \alpha)}, \varphi_2 \le \varphi \le \varphi_3
\end{aligned}
\tag{3.8}
$$

Figure 3.8 Feasibility analysis of the tool end



3.3.3 Feasibility analysis of the tool body

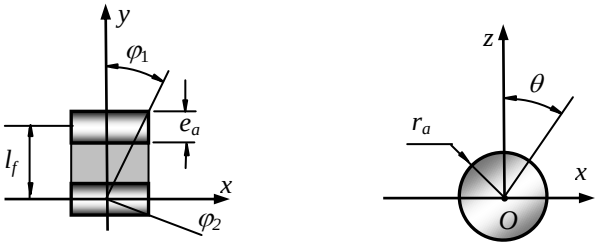
The tool body's use volume is a cylinder of radius r_a, and height e_a+l_f due to the tool body's translation l_f in the fastener removal direction. Two dividing angles φ_1, φ_2 are defined in Equation (3.9) in the searching space. The interference checking is executed using Equation (3.10) at both cylindrical and y-direction based on $GAC^{d}_{(R)}$. The depth $R(\theta, \varphi)$ at a direction of $GAC^{d}_{(R)}$ is projected on xz-plane and compared with r_a to check the cylindrical condition. Figure 3.9 illustrates the tool body's feasibility analysis.

$$
\begin{aligned}
&\varphi_1 = \tan^{-1}\left(\frac{r_a}{0.5e_a + l_f}\right) \\
&\varphi_2 = \pi - \tan^{-1}\left(\frac{r_a}{0.5e_a}\right)
\end{aligned}
\tag{3.9}
$$

$$F=\begin{cases}1 \quad if\ (0\le\varphi\le\varphi_1)\cap[\left\|R(\theta,\varphi)\hat{\mathbf{T}}(\theta,\varphi)\cdot\hat{\mathbf{j}}\right\|-(0.5e_a+l_f)>0] \\ and \quad if\ (\varphi_1<\varphi<\varphi_2)\cap[\left\|R(\theta,\varphi)\hat{\mathbf{T}}(\theta,\varphi)\cdot(\hat{\mathbf{i}}+\hat{\mathbf{k}})\right\|-r_a>0] \\ and \quad if\ (\varphi_2\le\varphi\le\pi)\cap[\left\|R(\theta,\varphi)\hat{\mathbf{T}}(\theta,\varphi)\cdot\hat{\mathbf{j}}\right\|-0.5e_a>0] \\ 0 \quad otherwise\end{cases} \tag{3.10}$$

$(0\le\varphi\le\pi \text{ and } 0\le\theta\le 2\pi)$

Figure 3.9 Feasibility analysis of the tool body



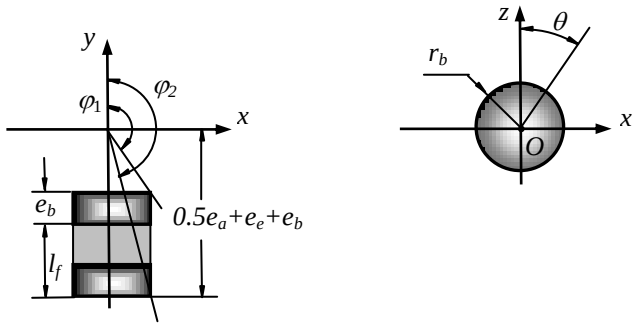
3.3.4 Feasibility analysis of the tool head and tool extension

The tool head's use volume is a cylinder of radius r_b, height e_b+l_f due to the tool head's translation l_f in the fastener removal direction. Two dividing angles φ_1, φ_2 are defined in Equation (3.11) in the search space. The checking process is performed based on Equation (3.12) along both directions φ and θ in the accessibility map of a $GAC^{d}_{(R)}$. Figure 3.10 illustrates tool head's feasibility analysis.

$$\begin{aligned}\varphi_1&=\pi-\tan^{-1}\left(\frac{r_b}{0.5e_a+e_e-l_f}\right)\\ \varphi_2&=\pi-\tan^{-1}\left(\frac{r_b}{0.5e_a+e_e+e_b}\right)\end{aligned} \tag{3.11}$$

$$F=\begin{cases}1 & \text{if } (\varphi_1 \le \varphi < \varphi_2) \cap [\left\| R(\theta,\varphi)\hat{\mathbf{T}}(\theta,\varphi)\cdot\left(\hat{\mathbf{i}}+\hat{\mathbf{k}}\right)\right\| - r_b > 0] \\ \text{and} & \text{if } (\varphi_2 \le \varphi \le \pi) \cap [\left\| R(\theta,\varphi)\hat{\mathbf{T}}(\theta,\varphi)\cdot\hat{\mathbf{j}}\right\| - (0.5e_a + e_e + e_b) > 0] \\ 0 & \text{otherwise}\end{cases} \tag{3.12}$$
$$(\varphi_1 \le \varphi \le \pi \text{ and } 0 \le \theta < 2\pi)$$

Figure 3.10 Feasibility analysis of the tool head

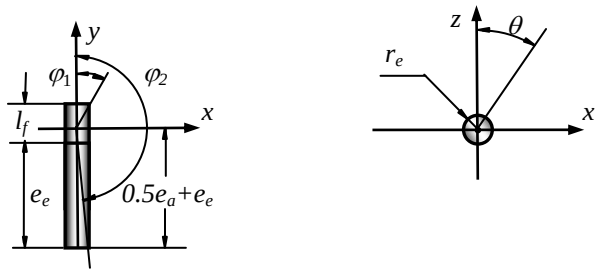


Similarly, the tool extension's dividing angles and checking criteria can be expressed in Equations (3.13) and (3.14). Since the upper and lower boundaries of the tool extension are covered by the tool body and the tool head respectively, only cylindrical condition is checked. Figure 3.11 illustrates the tool extension's feasibility analysis.

$$\varphi_1 = \tan^{-1}\left(\frac{r_e}{l_f - 0.5e_a}\right)$$
$$\varphi_2 = \pi - \tan^{-1}\left(\frac{r_e}{0.5e_a + e_e}\right) \tag{3.13}$$

$$F=\begin{cases}1 & \text{if } \left\| R(\theta,\varphi)\hat{\mathbf{T}}(\theta,\varphi)\cdot\left(\hat{\mathbf{i}}+\hat{\mathbf{k}}\right)\right\| - r_e > 0 \\ 0 & \text{otherwise}\end{cases} \tag{3.14}$$
$$(\varphi_1 \le \varphi \le \varphi_2 \text{ and } 0 \le \theta < 2\pi)$$

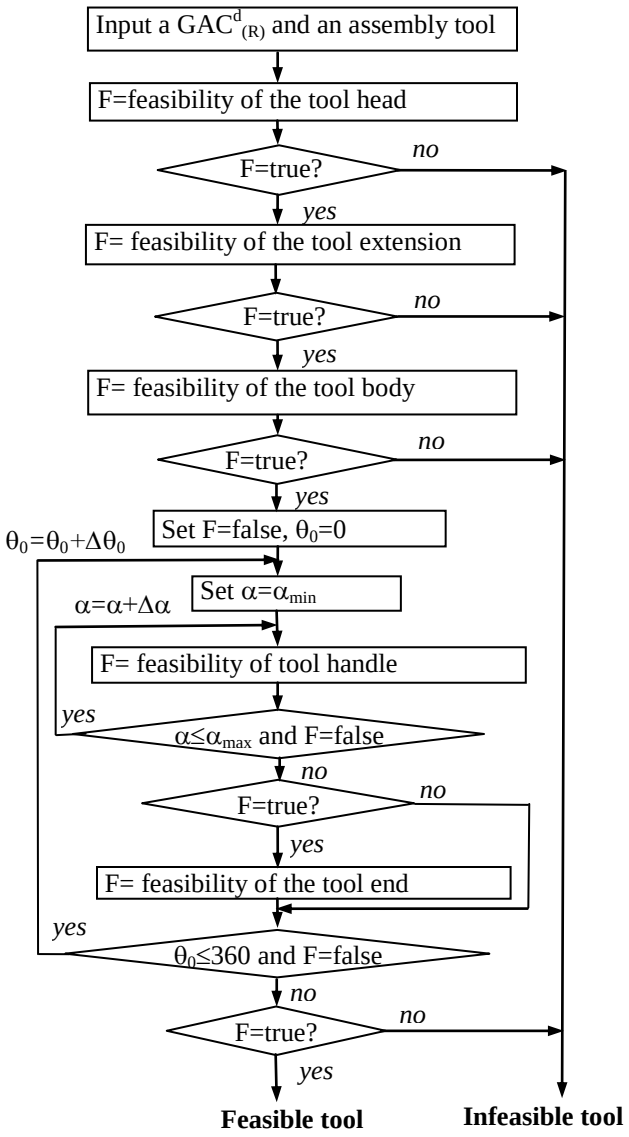
Figure 3.11 Feasibility analysis of the tool extension



3.3.5 Overall process of tool feasibility analysis

Figure 3.12 shows the flow chart describing the overall process of tool feasibility analysis. For a given fastener and an applied assembly tool, the $GAC^{d}_{(R)}$ is generated around the tool joint related to the fastener. Based on the $GAC^{d}_{(R)}$ information and the assembly tool's parameters, a searching process is executed to determine its feasibility. First, the tool head's use volume is checked for interference. If it collides with the surroundings, the tool is infeasible and the process is terminated. Otherwise, a similar interference checking for tool extension and tool body is executed. Finally, the feasibility analysis for the tool handle and the tool end starts to search for a collision-free tool placement at longitude angle θ_0 and tool access angle α. If the tool is infeasible at the tool placement (θ_0, α), the longitude angle θ_0 and tool access angle α are increased respectively for the testing of next tool placement. This process continues until an interference-free tool placement position is found. When all the five tool parts are feasible, the assembly tool is determined to be feasible.

Figure 3.12 Overall process of tool feasibility analysis



3.4 Implementation results

An assembly tool feasibility analysis system for fixture assembly planning is implemented via Web and Java DB connectivity (JDBC) technologies. Figure 3.13 (a) shows the tool feasibility test window for a wrench applied on a fastener. The fixture and its

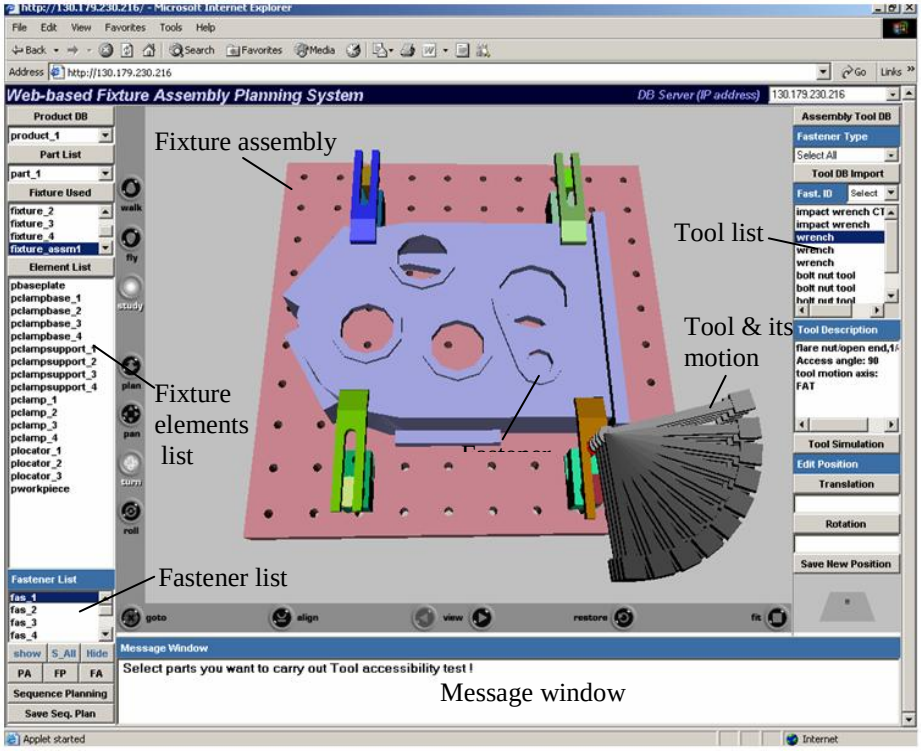
element information are shown in the left side. The fixture elements are retrieved from the fixture database, and the models are displayed in the Cortona browser. Available tools matching the fastener are chosen from the tool database and list in the right side. The tool and its motion are simulated in the browser window.

To simulate the tool feasibility analysis, the fixture elements and fasteners are selected for construction of $GAC^{d}_{(R)}$. The fixture elements included in the subassembly are changing during a fixture assembly process. Figure 3.13 (b) illustrates tool parameters and the 2D accessibility map of $GAC^{d}_{(R)}$, as well as the measured CPU time. The 2D accessibility map of $GAC^{d}_{(R)}$ is shown in the upper part of Figure 3.13 (b), in which the black area represents the blocked directions, and the white area represents the feasible moving directions. The assembly tool model is shown in the central diagram. The lower part in Figure 3.13 (b) is the value of tool parameters used for the tool feasibility analysis. The lowest textbox shows the result indicating the feasibility of an assembly tool, and the computing time for tool reasoning. In this case, it takes 12.343 seconds to create $GAC^{d}_{(R)}$ with line length R=161 mm. The tool reasoning time is 0.156 seconds. They are measured on a Pentium 4 2.4GHz PC with 1G RAM.

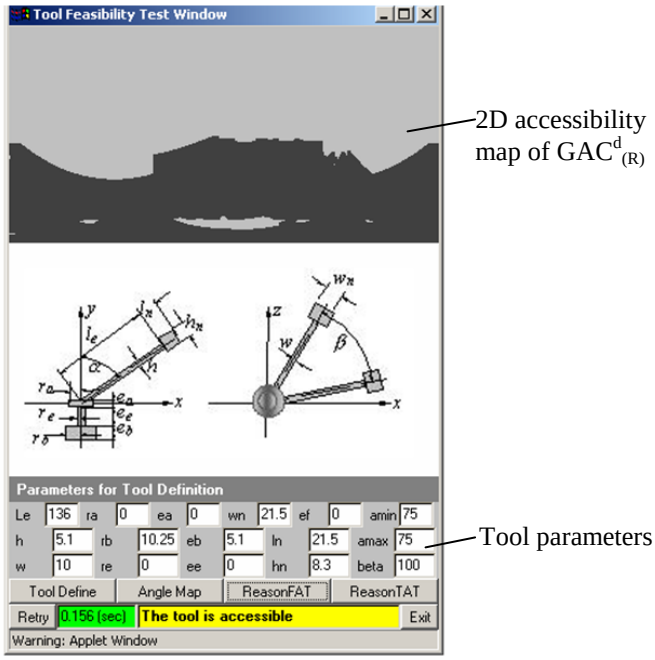
Similar to Figure 3.13 (b), Figure 3.14 shows other examples of assembly tools' feasibility analysis results. A wrench with an interchangeable socket to fit the bolt is analyzed in Figure 3.14(a). It takes 5.844 seconds to generate $GAC^{d}_{(R)}$ and the tool reasoning time is 0.156 seconds. A power wrench is simulated in Figure 3.14(b). It takes 0.719 seconds for generating $GAC^{d}_{(R)}$ and 0.094 seconds for tool reasoning. An open-end wrench applied on fastener 5 is evaluated in Figure 3.14(c). It indicates that this is an infeasible tool

blocked by the element 1. It takes 9.562s to generate $GAC^{d}_{(R)}$ and 0.14 seconds for tool reasoning. A speeder with an extension and a deep socket is evaluated in Figure 3.14(d). It is also an inaccessible tool blocked by the element 2. It takes 10.641 seconds for generating $GAC^{d}_{(R)}$ and 0.062 seconds for tool reasoning.

Figure 3.13 Assembly tool feasibility test windows (a) tool feasibility test window (b) tool parameters and 2D accessibility map of $GAC^{d}_{(R)}$

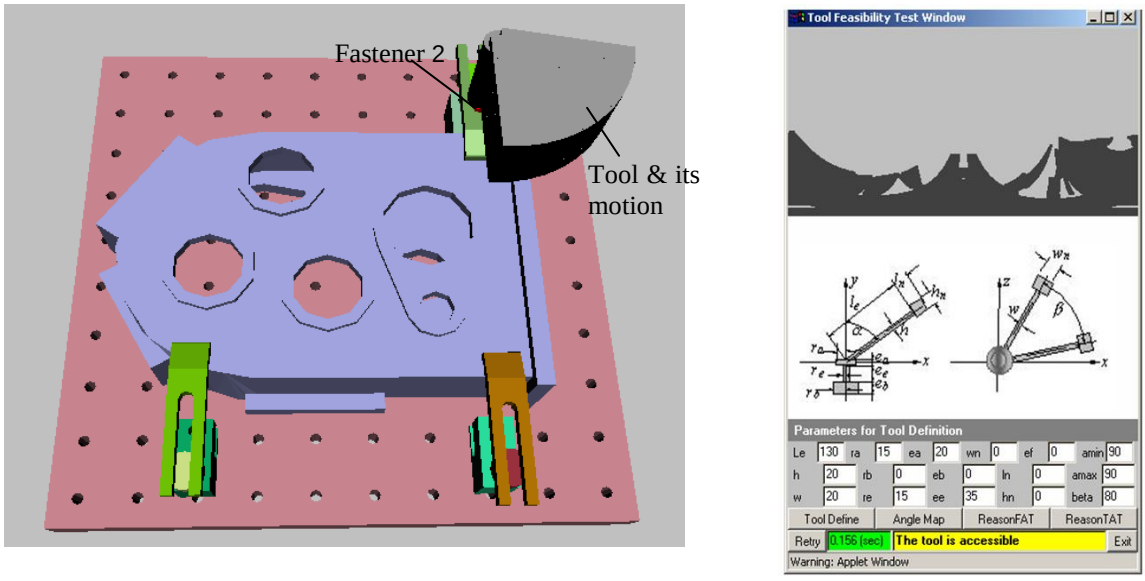


(a)

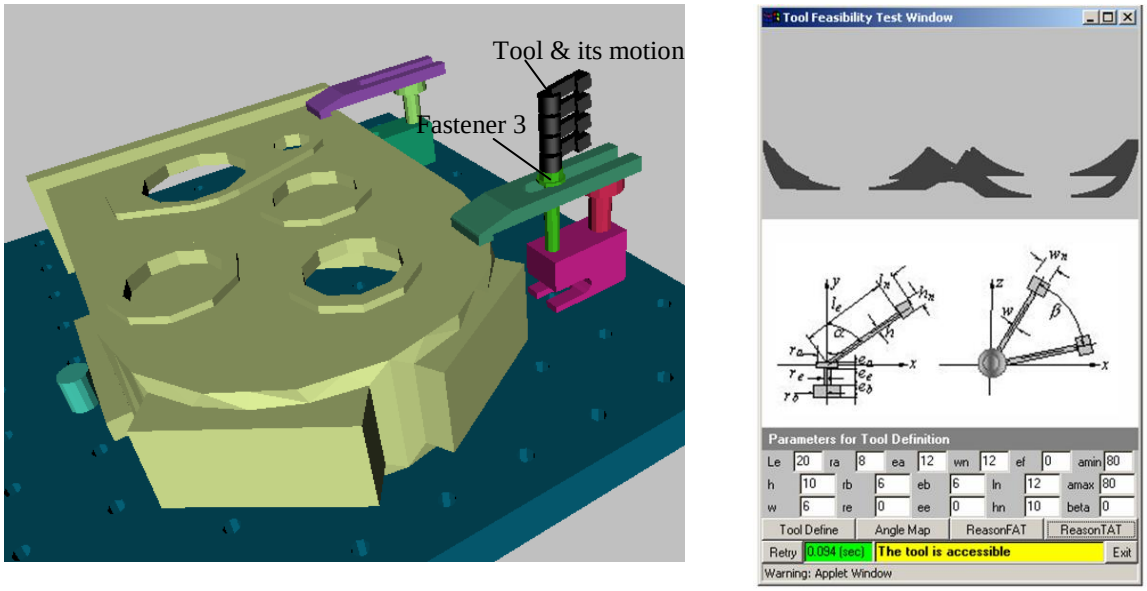


(b)

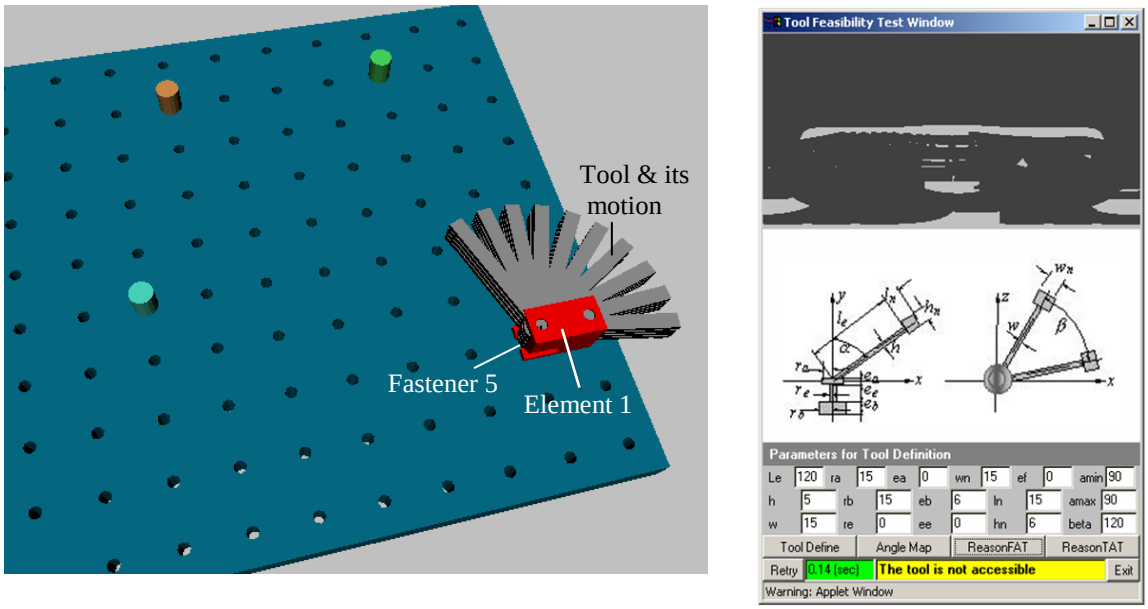
Figure 3.14 Feasibility test of several assembly tools (a) a wrench with socket (b) a power wrench (c) an open-end wrench (d) a speeder with an extension and deep socket



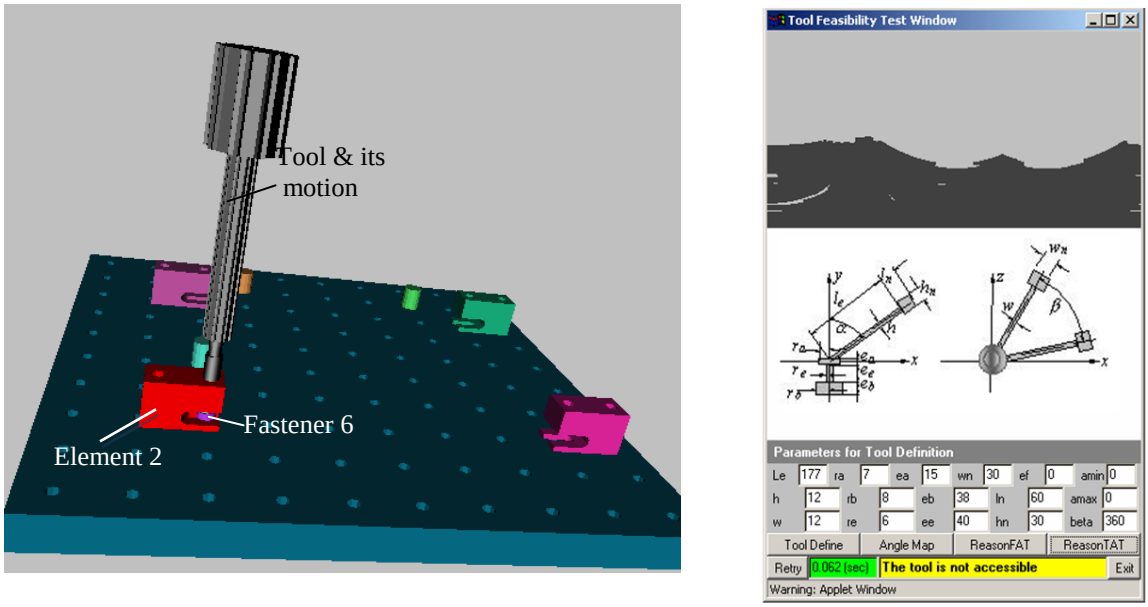
(a)


Tool & its motion
Fastener 3
Tool Feasibility Test Window
Parameters for Tool Definition
Tool Define
Angle Map
ReasonFAT
ReasonTAT
Retry
0.094 [sec]
The tool is accessible
Exit
Warning: Applet Window

(b)


Tool & its motion
Fastener 5
Element 1
Tool Feasibility Test Window
Parameters for Tool Definition
Tool Define
Angle Map
ReasonFAT
ReasonTAT
Retry
0.14 [sec]
The tool is not accessible
Exit
Warning: Applet Window

(c)



(d)

3.5 Chapter summary

Tooling consideration in fixture assembly planning often requires complex geometrical reasoning which is either unavailable or too computational expensive for industrial applications. This chapter presents a new approach for the tool feasibility analysis in fixture assembly planning. A new $GAC^{d}_{(R)}$ is defined to represent obstacles around a tool within the effective tool maximum length.

Assembly tools are represented as five articulated parts with seventeen parameters. The feasibility analysis is separately executed for the five parts based on their use volumes and $GAC^{d}_{(R)}$ information. The proposed method is advantageous as only boundaries of the tool use volume is required to reason instead of the whole boundaries of every tool applying position. The implementation illustrates the tool reasoning in fixture assembly

planning. The method developed can also support other applications such as Computer-aided Process Planning (CAPP), Design for Manufacturability (DFM), and Design for Assembly (DFA). In next chapter, the evaluation of assembly tool will be integrated into the fixture assembly planning to generate an optimum and feasible fixture assembly plan.

Chapter 4

Fixture assembly planning

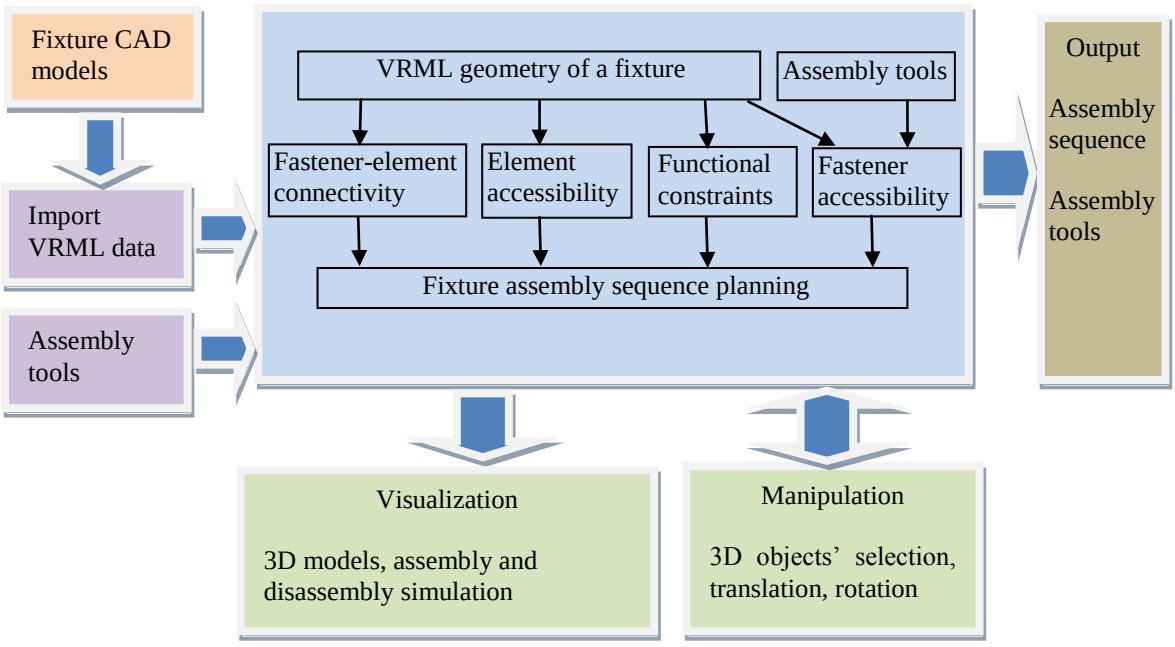
This chapter presents a fastener-based approach to fixture assembly planning in a Web-based environment. The developed approach generates a fixture assembly and disassembly plan and an assembly tool list based on the topological geometry constraints, the feasibility of assembly tools, and fixture functional constraints. It can reduce the lead time for fixture assembly and disassembly planning, and the selection of assembly tools. It can also be used to evaluate the feasibility of fixture configuration designs.

4.1 Framework of the Web-based fixture assembly planning system (WFAPS)

In fixture planning, a feasible assembly sequence is required for the fixture assembly based on the fixture design. Fixture disassembly is a process to separate a fixture into fixture elements and fasteners. In most cases, fixture disassembly is a reverse process of fixture assembly. The proposed system automatically generates a fixture assembly plan and feasible assembly tools' list in a distributed environment. A fastener-based assembly planning approach is proposed based on the topological disassemblability of fixture ele-

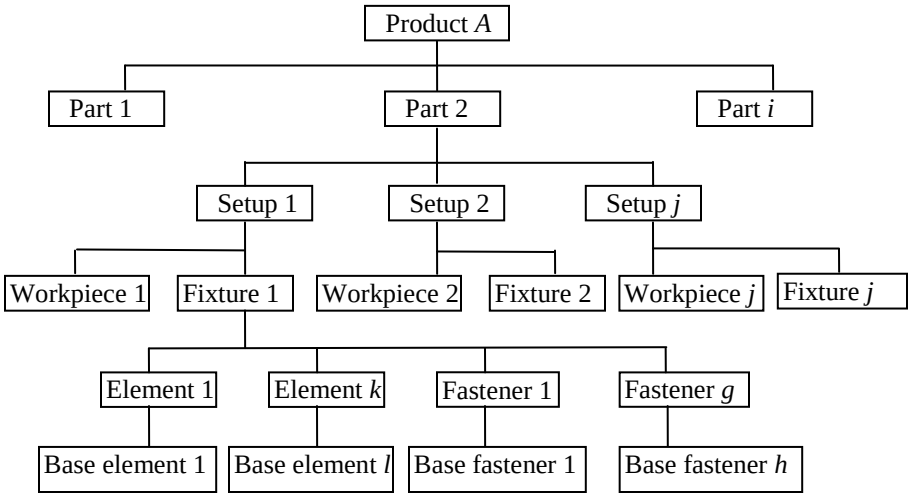
ments and assembly tool's feasibility applied on fasteners. Figure 4.1 shows the framework of WFAPS. The fixture models are generated in CAD systems and saved as Virtual Reality Modeling Language (VMRL) files from which geometrical information is extracted. The extracted VRML data are saved to a database (DB) server. The parameters of all assembly tools are also stored in the DB. The automatic assembly planning is executed based on the geometry reasoning of fixture elements and assembly tools. It analyzes the fastener and fixture elements' connectivity, fixture elements' accessibility, functional constraints of fixture elements, and fasteners' accessibility with respect to assembly tools. The fixture assembly sequence is generated based on the reasoning. The assembly sequence is simulated through a VRML browser. The system output includes the fixture assembly plan and an assembly tools list.

Figure 4.1 The framework of WFAPS



The data structure is mainly developed based on standard modular fixture. Reusable standard elements are commonly used in a modular fixture to construct a variety of fixtures for different workpieces. Modular fixture elements include a base plate, supports, locators, and clamps. The fixture assembly begins with the base plate which is fixed on the working table of a machine tool. The supporting, locating and clamping elements and workpiece are then mounted onto the base plate. Clamps are applied to hold the workpiece securely against the locators during machining operations. Some supports may be positioned under a workpiece to construct a locating plane. Other supports reinforce the stability of a workpiece to avoid possible displacement and excessive deformation.

Figure 4.2 The hierarchical structure of a product

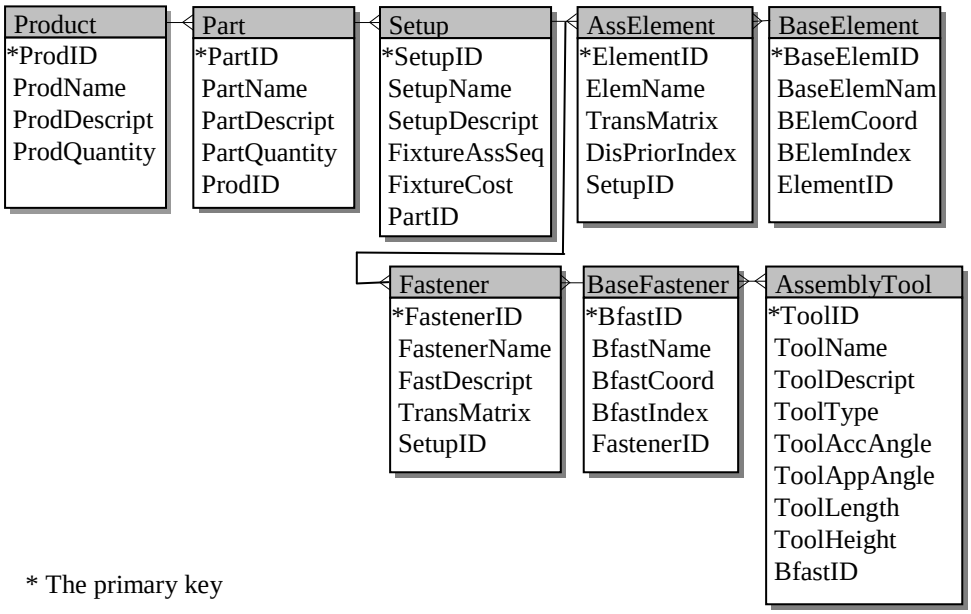


A product hierarchical structure is used to effectively manage and store the product and fixture data as shown in Figure 4.2. A product consists of some parts. Several setups may be required for each part to be machined, inspected or assembled. A fixture is normally

needed for each setup. Fixture assembly consists of a workpiece and a fixture. Fixture includes fixture elements and fasteners. The base elements and base fasteners are represented with original VRML geometry data before any transformation in Figure 4.2.

A relational database for fixture assembly planning is defined according to the hierarchical relationship of products and fixtures. Figure 4.3 shows the relational database model of WFAPS. Eight tables are used to store the product and fixture information, including the product, part, setup, assembly element, base element, fastener, base fastener, and assembly tool. The relationships of these tables are one-to many or many to many as shown in Figure 4.3.

Figure 4.3 The relational database model of WFAPS



4.2 A fastener-based approach to fixture assembly planning

Fixture assembly planning is subject to several constraints including geometrical constraints, fastener constraints, assembly tool constraints, and fixture functional constraints. The generation of a fixture assembly sequence is based on the disassemblability analysis of a fixture. The disassemblability of fixture elements can be evaluated according to the geometrical constraints of the fixture elements and fasteners. The fastener-based approach is an extension of the hybrid approach developed by Chung and Peng (2006-2). The CAD model of a fixture assembly is converted into VRML data as polygonized triangle patches. The fastener-element and element-element interference are tested between these patches using the hierarchical-oriented bounding box (OBB)-tree method and the triangle-triangle intersection method. Three primitive matrices, fastener-element connectivity matrix (*FP*), element accessibility matrix (*PA*) and fastener accessibility matrix (*FA*), are automatically generated based on the geometrical relationship among assembly elements and fasteners. *PA* defines the accessible disassembly directions of an element in a 2D digitalized chart. *FA* determines if a fastener can be assembled using an assembly tool without interference with other parts. In addition, the functional constraint of a fixture is considered which is different from general product assembly planning.

4.2.1 Geometrical constraints: FP and PA matrices

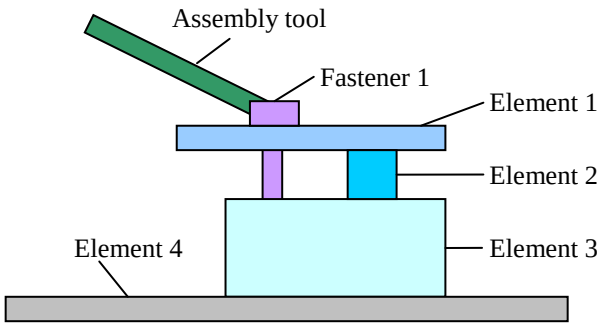
FP matrix is automatically determined to represent the fastener-element connection using collision detection methods. Assume the number of fasteners is *m*, and the number of as-

sembly elements is *n* in a fixture assembly. Let rows and columns of the matrix *FP* represent fasteners and assembly elements, respectively. Set $fp_{i,j}=1$ if fastener *i* is connected with element *j*, otherwise $fp_{i,j}=0$. *FP* is expressed in Equation (4.1) as follows.

$$FP = \begin{bmatrix} fp_{1,1} & fp_{1,2} & \cdots & fp_{1,n} \\ fp_{2,1} & fp_{2,2} & \cdots & fp_{2,n} \\ \cdots & \cdots & \cdots & \cdots \\ fp_{m,1} & fp_{m,2} & \cdots & fp_{m,n} \end{bmatrix} \tag{4.1}$$

Figure 4.4 shows an example of a fixture clamp tower. Since elements 1 is connected with element 3 by fastener 1, matrix *FP* = [1 0 1 0].

Figure 4.4 A fixture clamp tower



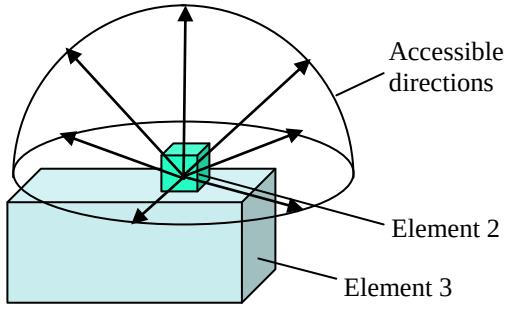
Similar to an assembly tree or an assembly graph, *PA* matrix represents the assembly relationship between elements. As shown in Equation (4.2), *PA* matrix is defined as the geometrical relationship of assembly elements. It is used to determine the accessible directions of element *i* relative to element *j*, in other words, the directions which part *i* can move along without blocked by part *j*, assuming that other obstacles in the environment are not considered.

$$PA=\begin{bmatrix} pa_{1,1} & pa_{1,2} & \dots & pa_{1,n} \\ pa_{2,1} & pa_{2,2} & \dots & pa_{2,n} \\ \dots & \dots & \dots & \dots \\ pa_{n,1} & pa_{n,2} & \dots & pa_{n,n} \end{bmatrix} \tag{4.2}$$

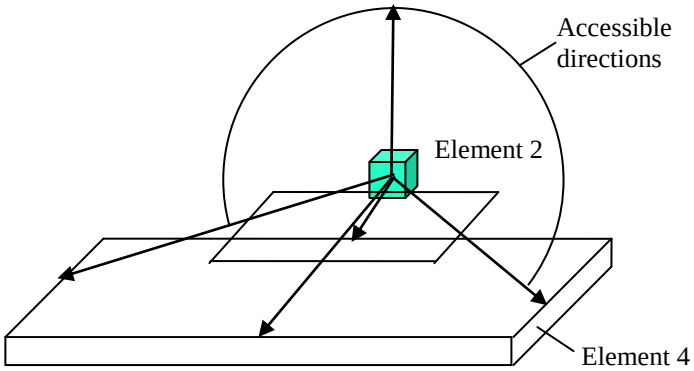
Take the fixture clamp tower in Figure 4.4 as an example. There are four fixture elements, that is n=4. Thus the *PA* matrix in Equation (4.2) becomes a *4x4* matrix. $pa_{2,3}$ in *PA* matrix represents the 3D accessible directions of element 2 relative to element 3 as shown in Figure 4.5 (a). Since element 2 contacts the up surface of element 3, the unblocked moving directions of element 2 are the directions from the centre of element 2 to any pixel in the up half sphere. $pa_{2,4}$ in *PA* matrix represents the 3D accessible directions of element 2 relative to element 4, which is shown in Figure 4.5 (b). The accessible directions cover a larger part of the sphere, compared to the up half sphere in Figure 4.5 (a). Figure 4.5 (c) shows the 2D accessible directions and their boundary formation. The boundary of accessible directions is the tangent lines of the edges on element 2 and element 4.

In a spherical coordinate system, these 3D directions can be represented as a 2D matrix or a 2D accessibility map using colatitude angles (φ) and longitude angles (θ). For each fixture element, the intersection of all accessible directions relative to other fixture elements forms the movable directions of this fixture element, so that there is no interference with any obstacle. For instance, for element 2 in Figure 4.4, the intersection of accessible directions of element 2 relative to other elements, including element 1, element 3, and element 4, forms the final accessible directions of element 2.

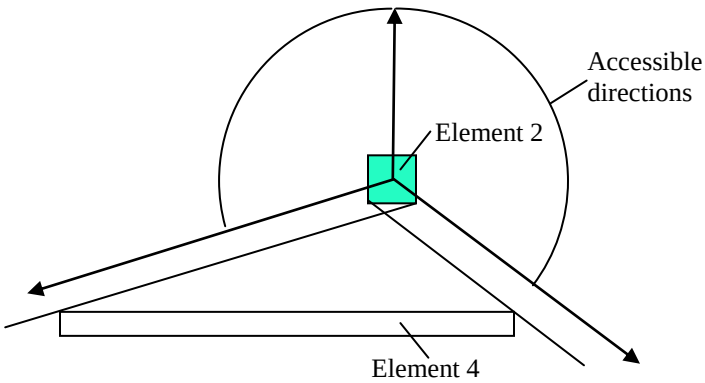
Figure 4.5 Accessible directions in PA matrix (a) Accessible directions of element 2 relative to elements 3 (b) Accessible directions of element 2 relative to element 4 (c) 2D accessible directions and their boundary formation of element 2 relative to element 4



(a)



(b)



(c)

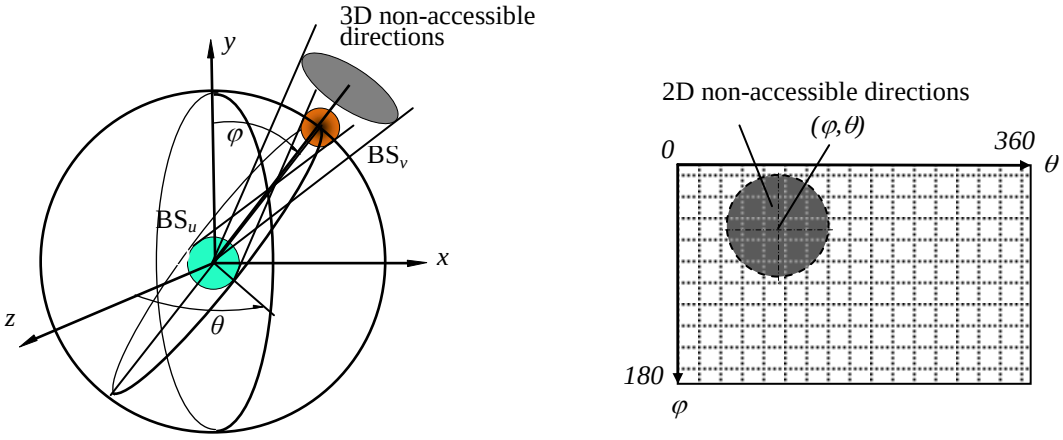
Each $pa_{i,j\ (i\neq j)}$ in PA can be expressed as a 180x360 matrix. An example of a $pa_{i,j}$ is shown in Equation (4.3). To automatically determine $pa_{i,j}$, a digitalized disassembly directionality chart (D^3C) is used to map all directions centred at element i (Chung and Peng, 2006-2). The chart is a 2D accessibility map representing 3D directions. The unit sphere surrounding an assembly element is discretized into 180x360 pixels in colatitude angles (φ) and longitude angles (θ) in a spherical coordinate system. The vectors from the sphere centre to the pixels form 180x360 directions. The value of a discrete pixel in the chart is set to 1 for an accessible direction, and 0 for an inaccessible direction.

$$pa_{i,j} = \begin{bmatrix} 1 & 1 & \dots & 0 & 0 & 0 & 0 & 1 & \dots & 1 \\ 1 & 1 & \dots & 0 & 0 & 0 & 0 & 1 & \dots & 1 \\ 1 & 1 & \dots & 0 & 0 & 0 & 0 & 1 & \dots & 1 \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ 1 & 1 & \dots & 1 & 1 & 1 & 1 & 1 & \dots & 1 \end{bmatrix} \quad (\varphi = 1, 2, \dots, 180,\ \theta = 1, 2, \dots, 360) \tag{4.3}$$

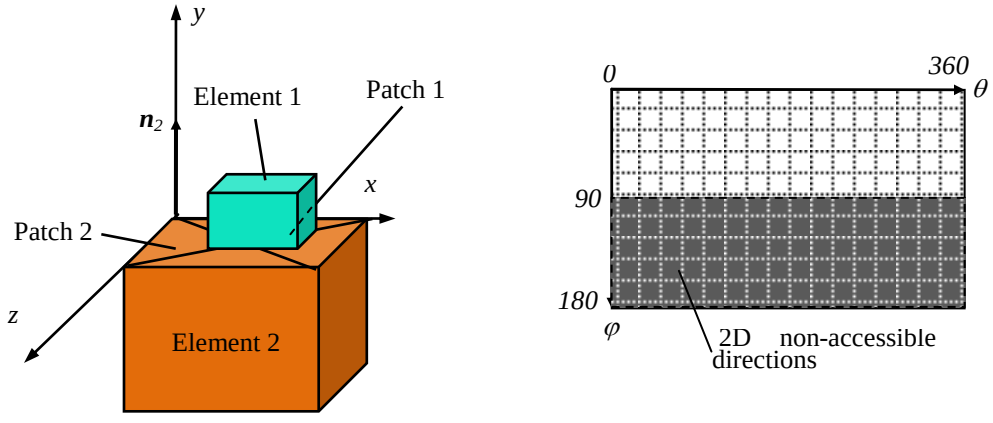
Figure 4.6 shows the generation of 2D D^3C from 3D accessible directions for two bounding spheres and two contact patches. In the collision detection process, an element may be divided as several oriented bounding boxes (OBB). A bounding sphere is an approximate sphere enclosing an element or an OBB. Assume that an element or its OBB is approximated by a bounding sphere BS_u, another element or its OBB is approximated by a bounding sphere BS_v. If these two bounding spheres, BS_u and BS_v, are separated, the 3D non-accessible directions form a cone. These directions can be mapped into a 2D D^3C as shown in the right side of Figure 4.6 (a). The non-accessible directions are then marked as grey color in the D^3C. If a contact is detected between two elements after recursive

collision detection, the non-accessible directions are marked. In Figure 4.6 (b), triangle patch 1 from an element 1 and patch 2 from another element 2 contact each other. If the surface normal of patch 2 is alone y axis, the non-accessible directions of the element 1 is a set of directions satisfying $90<\varphi\leq180$, $0<\theta\leq360$, as shown in D^3C in the right side of Figure 4.6 (b).

Figure 4.6 Generation of D^3C (a) mapping 3D accessible directions between two bounding spheres (b) mapping 3D accessible directions with contact surfaces



(a)



(b)

To disassemble a fixture element, two primary geometrical constraints should be fulfilled: there are accessible disassembly directions for the element, and there is no fastener connected to this element. The topological disassemblability (Δi) of element i within element set N is determined in Equation (4.4), where Δi =1 means the element can be disassembled.

$$\Delta i = \begin{cases} 1 & if \bigcap_{j=1,2,..n} pa_{i,j} \neq \Phi \\ 0 & otherwise \end{cases} \tag{4.4}$$

4.2.2 Assembly tool constraints: FA matrix

The accessibility of a fastener can be evaluated by finding a feasible assembly tool. A set of assembly tools and their parameters are stored in a tool DB and retrieved for tool accessibility analysis. As shown in Equation (4.5), the matrix *FA* represents if a fastener can be assembled by an assembly tool without interference with other elements. Let the rows and columns in *FA* refer to the fasteners and elements respectively. If an assembly tool applied to fastener i does not interfere with part j, then $fa_{i,j}$ =*0*. Otherwise, $fa_{i,j}$ =1. For example, in Figure 4.4, *FA* = [0 0 0 0] because the assembly tool applied on fastener 1 does not interfere with fixture elements 1, 2, 3, and 4. *FA* is determined by reasoning tool placement constraints and the tool use volume using a global accessibility cone with depth of a truncated half-line ($GAC^{d}_{(R)}$) method introduced in Chapter 3.

$$FA = \begin{bmatrix} fa_{1,1} & fa_{1,2} & \dots & fa_{1,n} \\ fa_{2,1} & fa_{2,2} & \dots & fa_{2,n} \\ \dots & \dots & \dots & \dots \\ fa_{m,1} & fa_{m,2} & \dots & fa_{m,n} \end{bmatrix} \tag{4.5}$$

The topological accessibility (λ_i) of fastener i within the fastener set M is expressed in Equation (4.6), where λ_i =1 means fastener i can be assembled with a feasible assembly tool, λ_i =0 indicates that the applied assembly tool interference with at least one element of the fixture.

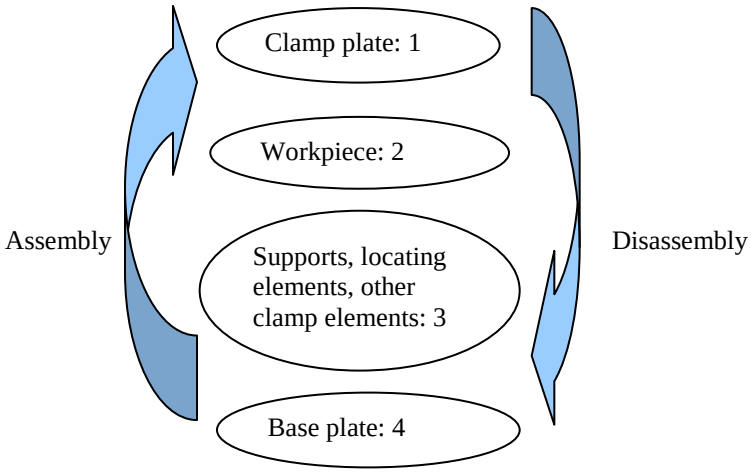
$$\lambda_i = \begin{cases} 1 & if \sum_{j=1}^{n} fa_{i,j} = 0 \\ 0 & otherwise \end{cases} \tag{4.6}$$

4.2.3 Functional constraints: disassembly priority index

Unlike an ordinary product assembly, the fixture assembly is subject to functional constraints related to the base plate, locators and clamps. For example, a workpiece can't be removed under an applied clamp force even if it has feasible disassembly directions. The fixture assembly consists of two steps: fixture elements assembly and workpiece loading. The fixture elements assembly is to construct the fixture including the base plate, locating elements, fixture supports, fixture clamps and relative fasteners. The workpiece loading is to position the workpiece and apply clamp force on it. After a workpiece is machined, it will be unloaded from the fixture. A batch of workpieces will be loaded and unloaded repeatedly on the same fixture for machining. A specific assembly sequence should be fol-

lowed to meet the functional constraints. Since a base plate forms the base of other assembly elements, it is first assembled to the working table of a machine tool. Although a base plate may be geometrically feasible to be assembled later, it should always be fixed at the beginning of the assembly in practice. Then, fixture locating elements can be assembled to ensure a precise locating position for a workpiece. Following the locating elements, the workpiece can be positioned and oriented in the fixture. Finally, the clamp elements are installed and the clamp force is applied on the workpiece. Here the clamp elements refer to the elements applied clamp force on the workpiece, such as a clamp plate. Other clamp elements may be assembled before a workpiece is assembled.

Figure 4.7 The disassembly priority index of assembly elements



A disassembly priority index (*DPI*) is proposed for fixture assembly planning to represent functional constraints of the fixture elements. The *DPI* value is assigned to fixture elements based on their disassembly priority. Low DPI indicates higher disassembly priority

while high *DPI* means lower disassembly priority. The *DPI* of the clamp plates is set to 1. The *DPI* of a workpiece is set to 2. The *DPI* of the supports, other clamp elements, and locating elements are set to 3. The *DPI* of the base plate is set to 4. Figure 4.7 shows the *DPI* of the assembly elements.

4.2.4 Fixture assembly sequence planning

There are two steps to generate an assembly sequence plan: rating disassembly levels for the fixture elements, and sequence planning. The element disassembly level matrix (*DL*) forms a predefined precedence graph of disassembly levels used to search for the assembly sequence. It has the same dimension with *PA*. A disassembly level is assigned to an assembly element if it satisfies three conditions: the element can be geometrically disassembled, the fastener connected to this element is accessible by an assembly tool without interference, and it fulfills the functional constraints. These conditions are expressed in Equation (4.4) for the disassemblability of an assembly element, in Equation (4.6) for the fastener accessibility, and in Figure 4.7 for functional disassembly priority constraints.

To rate disassembly levels, the assembly elements with low *DPI* are first evaluated to be disassembled. For each element in this *DPI* group, the topological disassemblability Δ_i is first determined. If there are fasteners connected to this element, the assembly tool feasibility and the fastener accessibility test is performed. When all elements of a low *DPI* group have been assigned a disassembly level, the assignment to next *DPI* group begins. In particular, elements in a *DPI* group may be assigned to several disassembly levels

based on the geometrical constraints. The disassembly level is registered in the triangle element of *DL*.

Figure 4.8 shows a flowchart to determine matrix *DL*. *M* stands for the fastener set in a fixture, and *N* represents an assembly element set. G_{DP} *(dpi)* is an element set whose *DPI* equals *dpi*. DPI_{max} is the maximum *DPI* value. First, three primitive matrices *PA*, *FP*, *FA* are generated automatically by geometrical reasoning. The topological disassemblability Δi of an assembly element in G_{DP} *(dpi)* with the lowest *DPI* is calculated. If Δi =1 for an element *i* in this group, the element *i* can be disassembled along a certain direction. Then the fastener set *Ft*, which is connected to the element *i*, is searched via *FP* matrix. For each fastener in *Ft*, an assembly tool is searched in the tool DB and tested for its feasibility. Once a suitable assembly tool is found, which means the fastener accessibility λ_j =1, a disassembly level is assigned to the element *i*. This element is added to the element set *Ps* and removed from G_{DP} *(dpi)*. Fastener set *Ft* is added to fastener set *Fs*. The free elements in G_{DP} *(dpi)* without connecting to any fastener are also assigned to this disassembly level. If any one of the two conditions fails, element *i* remains in the G_{DP} *(dpi)* and will be checked for next disassembly level. This process continues until all elements in G_{DP} *(dpi)* have been assigned a disassembly level. The elements in the next higher *DPI* group will form new disassembly levels. At the same time, the left element set, fastener set and disassembly level are updated accordingly. The program recursively checks the disassembly constraints of assembly elements and fasteners until all elements are assigned a disassembly level in *DL*.

Figure 4.8 A flowchart for the determination of matrix *DL*

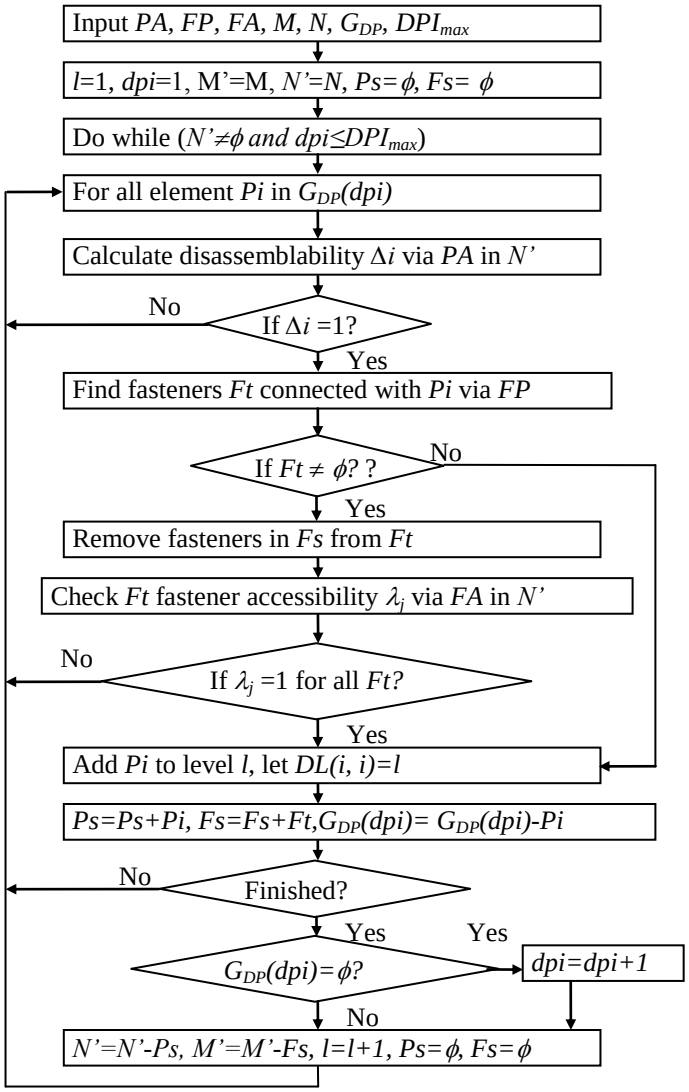
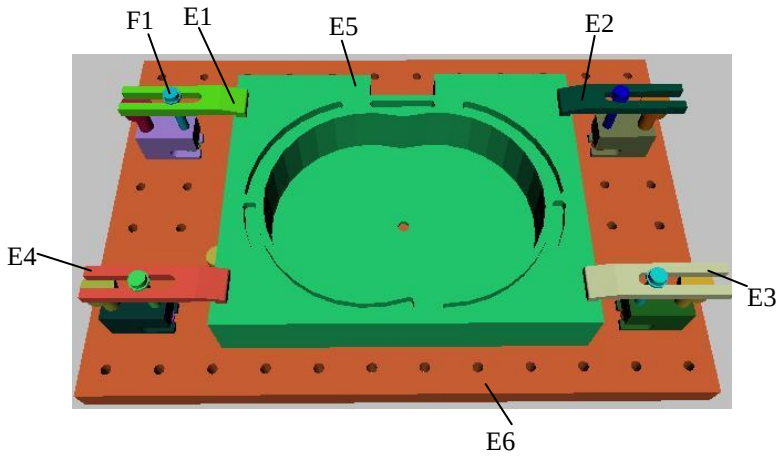


Figure 4.9 shows an example for generating disassembly level matrix *DL*. According to the disassembly priority index, clamp plates E1, E2, E3, and E4 have disassembly priority index *DPI*=1, so these four elements form a group $G_{DP}(dpi=1)$. In this group, each element is checked to see if it can be disassembled. If E1 is feasible to be disassembled,

its connecting fastener F1 is then checked. If there is a feasible assembly tool to disassemble F1, element E1 is assigned as disassemble level *l*=1. If E1 is not able to be disassembled or there is no feasible assembly tools for F1, E1 will remain in the group and be checked in the next disassembly level. Assuming that workpiece E5 has a disassembly level *l*=3, and base plate E6 has a disassembly level *l*=4, the generated disassembly matrix is shown in Equation (4.7). The sequence of rows and columns in the matrix *DL* is from E1 to E6.

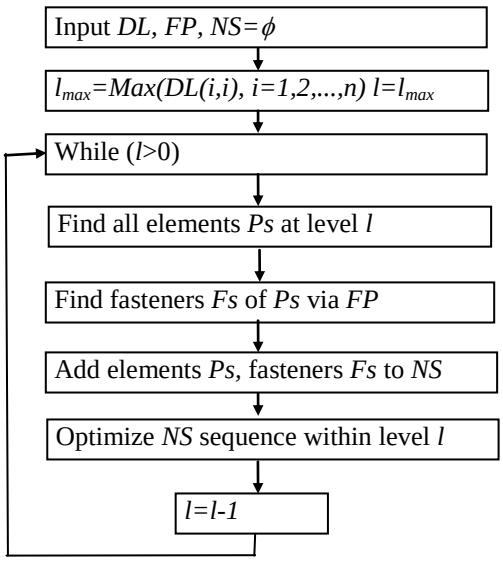
Figure 4.9 An example for generating disassembly level matrix *DL*



$$DL = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ & & 1 & 0 & 0 & 0 & 0 & 0 \\ & & & 1 & 0 & 0 & 0 & 0 \\ & & & & \ldots & 0 & 0 & 0 \\ & & & & & \ldots & 0 & 0 \\ & & & & & & 3 & 0 \\ & & & & & & & 4 \end{bmatrix} \tag{4.7}$$

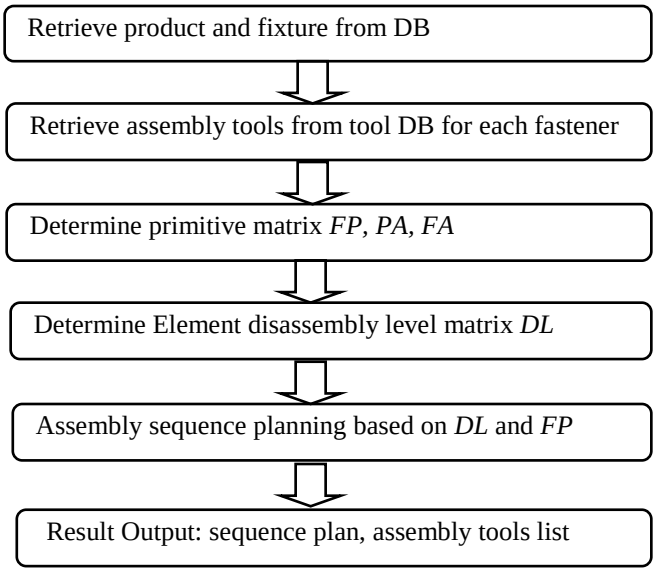
Generally speaking, the fixture assembly is a reverse process of fixture disassembly with the geometrical and functional constraints fulfilled. The fixture assembly planning begins with the highest disassembly level in *DL*. Figure 4.10 shows a flowchart for the fixture assembly planning based on matrix *DL* and *FP*. Let *NS* represents the assembly sequence. First, the elements *Ps* at the highest disassembly level are determined by *DL*. The connecting fasteners *Fs* are found via *FP*. *Ps* and *Fs* are added to *NS*. Then the next higher disassembly level are found and added to *NS*. This process continues until all elements and fasteners have been added to *NS*. For elements at the same disassembly level in *DL*, the assembly sequence is optimized by taking stability into consideration. Since fixture elements will be released due to gravitational forces acting on them, it is preferable to first disassemble the elements with accessible directions pointing upwards to keep the subassembly stable (Huang and Huang, 2002). In addition, free elements are disassembled earlier than the elements fixed by a fastener.

Figure 4.10 A flowchart for generating an assembly sequence based on *DL*, *FP*



An overall procedure of the fixture assembly planning is shown in Figure 4.11. The product and fixture information, and available assembly tools are retrieved from a database. Three primitive matrices *FP*, *PA*, *FA* are first determined followed by the disassembly level assignment to all elements in *DL*. The fixture assembly plan is generated subsequently.

Figure 4.11 An overall procedure of the assembly sequence planning

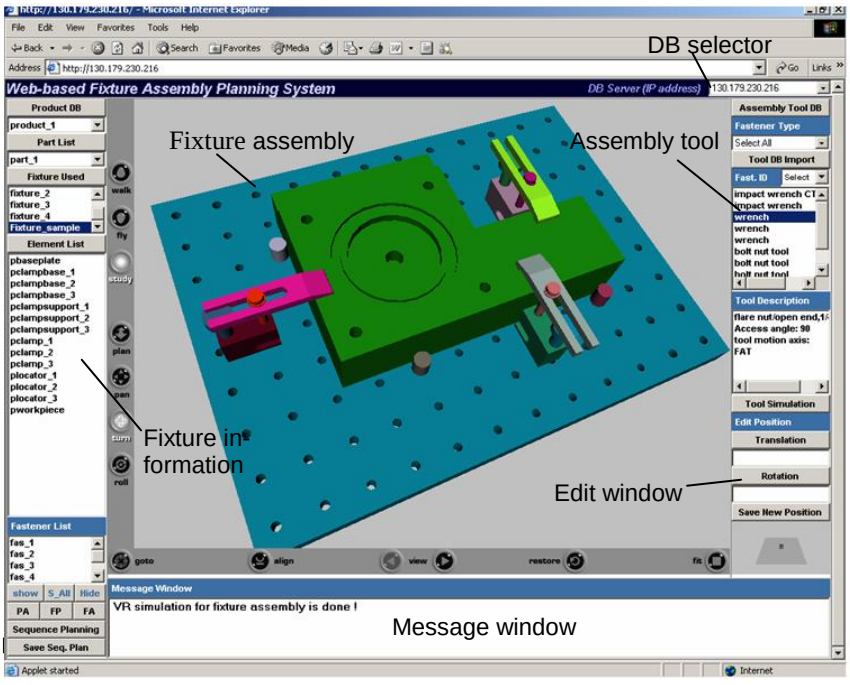


4.3 Implementation results

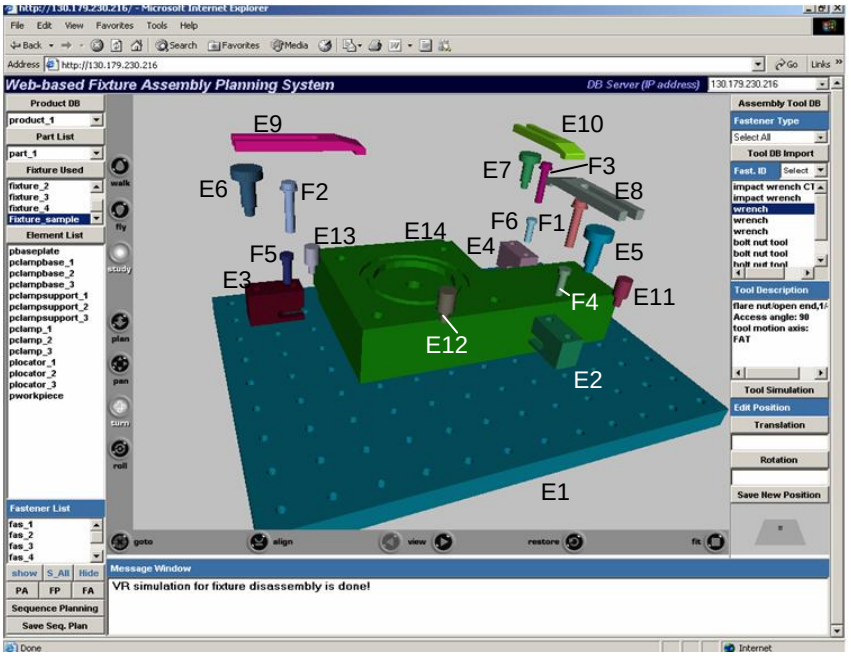
The WFAPS allows users to select a product from a product list in the DB. Corresponding part, fixture, and fixture elements, fasteners are interactively selected and retrieved from the DB based on the users' selection. Then the user can perform assembly planning step by step using the system interface. Figure 4.12 (a) shows the WFAPS's user interface. The selection and control buttons are located in the left side of the Webpage. The

original *PA*, *FP*, *FA* are generated automatically by clicking the corresponding selection. The fixture assembly sequence plan is generated by click the "Sequence Planning" button. In the right side of Figure 4.12 (a), the information of assembly tools is retrieved from the tool DB. The tools are then checked for the fastener accessibility. The position and orientation of the fixture elements may be adjusted and saved to the database if the assembly planning fails. The VRML browser is embedded in the middle of the window for the simulation of 3D fixture assembly models and assembly process.

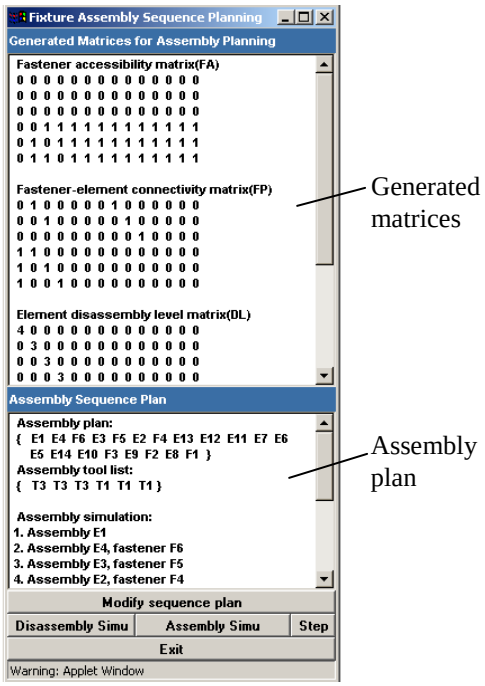
Figure 4.12 An example of the implemented WFAPS (a) user interface of the fixture assembly planning (b) simulation of the fixture assembly/disassembly (c) Fixture assembly sequence plan window



(a)



(b)



(c)

Figure 4.12 (b) illustrates a simulation of the fixture assembly and disassembly. The elements move away along the feasible disassembly direction in the simulation according to

the generated disassembly plan, shown in Figure 4.12 (c). In Matrix *DL*, four fixture disassembly levels are generated based on the developed method. The clamp plate and workpiece are in the first and second disassembly levels respectively. Three locators, the clamp support, and the clamp base are in the third disassembly level. The base plate is in the last disassembly level. The assembly sequence plan is generated as "E1- E4- F6- E3- F5- E2- F4- E13- E12- E11- E7- E6- E5- E14- E10- F3- E9- F2- E8- F1". E1-E14 represents assembly elements shown in element list. F1-F6 represents fasteners shown in fastener list. The required assembly tools are also listed in the sequence plan window. The assembly plan can be saved in the fixture DB.

4.4 Chapter summary

An automatic fixture assembly planning tool is developed to support the collaboration of fixture design in a distributed environment. It helps to reduce the lead time for fixture assembly/disassembly planning and the assembly tool selection, as well as to verify the feasibility of a fixture assembly and disassembly in the design stage.

The developed fixture assembly planning approach is based on the analysis of topological disassemblability, assembly tool feasibility, and fixture functional constraints. The geometrical constraints of fixture elements are automatically reasoned based on the collision detection methods. The assembly tool feasibility is verified for each fastener. A fixture assembly sequence plan is generated with a feasible assembly tools list.

Chapter 5

Motion planning in fixture loading and unloading

This chapter presents a new accessibility-based A Star (A*) algorithm for motion planning in fixture loading and unloading. A spherical accessibility matrix is used for the representation of accessible moving directions. Based on the spherical accessibility matrix, randomly sampled accessible directions are integrated into A* algorithm to find an optimal moving path. The proposed algorithm has been implemented in a Web-based virtual environment for fixture loading and unloading. Simulation of fixture loading scenarios shows that the algorithm is efficient and able to generate collision-free optimal paths.

5.1 3D accessibility-based A* algorithm

The feasibility of a fixture loading path depends on a workpiece's geometry and its machining environment. A machining environment consists of the machine tool, fixture elements, and cutting tools. During workpiece loading, the workpiece and its holding de-

vices have to avoid any possible collision with these obstacles. If the workpiece is loaded manually, operator's hand serves as the holding device. If the loading operation is performed with the aid of a robot, a gripper could be the holding device. To simplify the problem, only the geometry of a workpiece is considered movable in this research. The obstacles are assumed to be stable over time.

In a product disassembly, many parts may be removed by a single translation along a line. The removing direction should satisfy a local detachment condition and being collision-free with other elements. Similar conditions are true for fixture loading. However, a buffer's location to store the workpiece is usually predetermined beside the machine tool used to process the workpiece. The workpiece's travelling distance from a buffer to a fixture is farther than that in disassembly. A single translation is most likely blocked by some obstacles. Thus, multiple translations are required in fixture loading. Moreover, since there are numerous possible moving paths, the optimal path should be the shortest one without unnecessary detour or rotation. A spherical accessibility matrix is proposed to represent feasible 3D moving directions for each intermediate moving position of a workpiece.

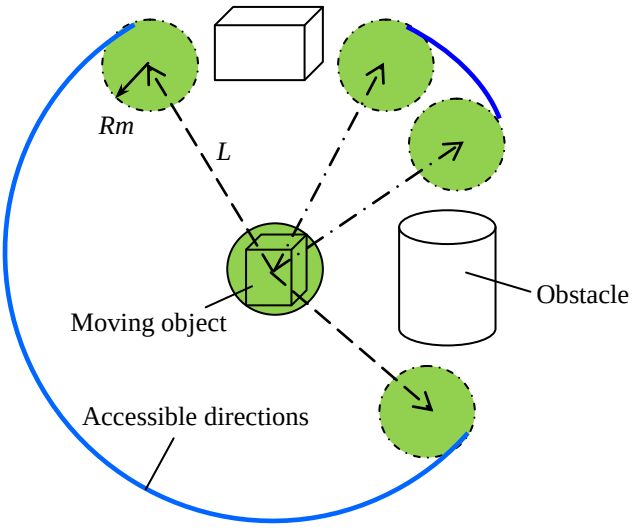
5.2 Construction of Spherical Accessibility Matrix

Similar to the accessibility matrix PA discussed in the previous chapter, the construction of spherical accessibility matrix in this research is based on a digitized disassembly directionality chart (D^3C) (Chung and Peng, 2006-2). As mentioned before, D^3C is a 2D chart representing the surrounding obstacles information in a set of directions. It stores the feasible moving directions for a moving object. 180x360 corresponding directions are

formed and equally distributed on the unit sphere. Each direction is represented as a Boolean value. True refers to inaccessible moving directions. False represents accessible directions to move.

The spherical accessibility matrix is a union of all 180x360 directions by mapping each obstacle to a workpiece. In fixture assembly planning, the moving distance is considered infinity. In other words, all the objects in the environment are included in the accessibility calculation, no matter how far their locations are. Different from fixture assembly planning, obstacles that are beyond a limited moving distance are excluded in this motion planning. If L is a one step moving distance of an object, it is clear that only the obstacles inside the L sphere may collide with the object. Inclusion of obstacles beyond L may lead to inefficient motion planning due to the geometrical reasoning of a large number of obstacles. In addition, accessible moving directions may be falsely assessed as inaccessible. Thus, the obstacles and triangle patches farther than a moving distance L should be excluded from the interference checking. If the distance L is measured from the center of the moving object, the geometry of the moving object should be taken into account. Assume the moving object is approximated by bounding sphere and its spherical radius is Rm, the obstacles farther than $L+ Rm$ are discarded from interference checking. Figure 5.1 shows an object's accessible directions marked in the arc.

Figure 5.1 An object's accessible moving directions



After the obstacles included in the calculation have been determined, the directional mapping between each obstacle and the workpiece are formed. For two objects with separated bounding sphere, the blocking area is a round area. For closer objects, mapping is based on recursive oriented bounding box (OBB) division. If a contact is detected, a triangle patch-based mapping is executed. The spherical accessibility matrix can be expressed as:

$S=\{s(L, \varphi, \theta)$ = true or false, where φ=1, 2,……180; θ=1, 2, ……360$\}$

where $s(L, \varphi, \theta)$ represents the accessibility of a moving direction within L distance in 3D space.

5.3 Accessibility-based A* algorithm

The proposed accessibility-based A* algorithm is an incremental sampling and searching approach. The search space is represented by a search tree which is incrementally constructed by a sequence of sampling positions of a workpiece. The tree is not extended to the goal at one step but to points at a random direction with limited distance from previous position. The moving direction is randomly generated by linking the current position with a new sample position. A new sample position is valid if the moving direction is accessible based on the spherical accessibility matrix.

Figure 5.2 The generation of moving directions

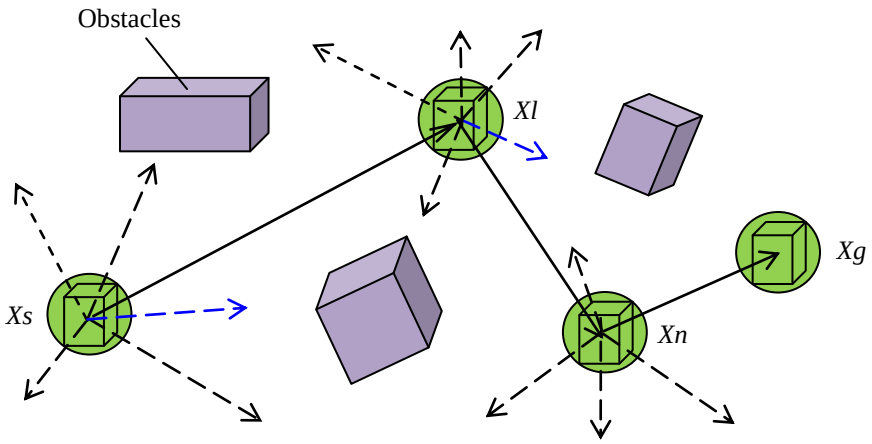


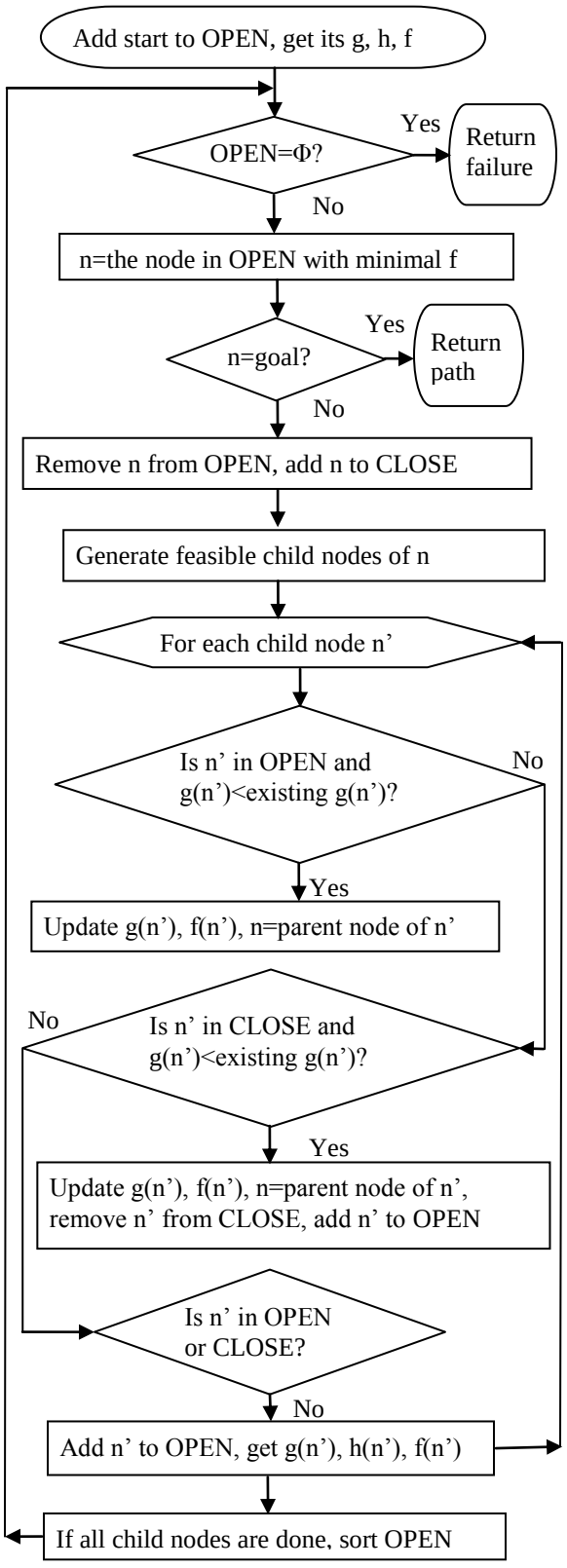
Figure 5.2 shows the generation of moving directions. *Xs* is the starting position, *Xg* is the goal position. The path planning is to find a path linking *Xs* and *Xg* without interference with the obstacles. The path may consist of several line segments if a simple path doesn't exist. The path planning starts with start position *Xs*. Several random positions are generated within a predetermined moving distance *L*. Assuming a random position is *Xl*, the

line segment *XsXl* forms a possible moving direction. If *XsXl* is accessible based on spherical accessibility matrix, *Xl* is accepted and evaluated by its cost function which is the total distance from start to goal via *Xl*. All inaccessible positions are discarded.

The single-tree search is similar as adding the goal to the tree. When a new node is added to the tree, the node is evaluated by a cost function. The objective is to minimize the distance from start to goal. Assume *OPEN* list stores new unvisited nodes (position). *CLOSE* list stores visited nodes. *NEIGHBOR* list stores generated accessible child nodes of node *n*. *L* is the maximal moving distance per step. *g(n)* is the cost from start to node *n*, *h(n)* is the heuristic estimate cost from *n* to goal. The total cost *f(n)=g(n)+h(n)*.

Figure 5.3 shows the flowchart of the accessibility-based A* algorithm. The search starts by adding *start* to *OPEN*. While *OPEN* is not empty, node *n* in *OPEN* with minimal cost is selected for expansion. Feasible child nodes of *n* are generated with three steps. First, a spherical accessibility matrix is calculated at node *n*. Then random moving directions (φ, θ) from *n* are generated and the accessibilities of these directions are evaluated based on the matrix. Finally, accessible moving positions are added to *NEIGHBOR* as child nodes of *n*. For each child node of *n*, if it is already in *OPEN* or *CLOSE* and new cost *g(n')* is smaller than the existing cost *g(n')*, the cost and its parent node is updated. Note that both new and existing paths lead to this node. If a child node is not in *OPEN* or *CLOSE*, it is added to *OPEN*. After all child nodes are added to the tree, *OPEN* is sorted to select the best node for next expansion. Such search and expansion repeats recursively until the goal is reached or search fails. To improve the searching efficiency, a node with the

Figure 5.3 Flowchart of the proposed accessibility-based A* algorithm



direction towards the goal is repeatedly added as a child node. If the distance from node *n* to the goal is less than *L*, the goal is added as a child node of the node *n*.

5.4 Implementation results

The proposed accessibility-based A* algorithm is implemented in a Web-based virtual environment to simulate the fixture loading and unloading process. The workpiece, fixture, and environmental factors, such as the machine tool, are modeled and assembled in Pro/Engineer. The fixture elements and other environmental factors are assumed as obstacles in the motion planning. The models are retrieved from a DB through the Web server. Figure 5.4 shows a computer numerical control (CNC) milling machining center. It includes a work table, a spindle, two safety doors, etc. A virtual CNC machining center is constructed based on this machine for testing the proposed motion planning algorithm.

Figure 5.4 A real CNC milling machining center

Figure 5.5 shows the generated models and developed Web-based fixture motion planning system. The 3D models in the machining environment are illustrated in the middle of the Web page. The CNC machining center's body, its worktable, spindle, and doors are modeled to simulate the environmental factors. A buffer is modeled to store workpieces. Based on the retrieved 3D models, the 3D spherical accessibility matrix at different positions is generated. The A* algorithm begins to search for the shortest feasible path according to the spherical accessibility matrix. Different locations of the workpiece between the fixture and the buffer simulate the generated motion plan in fixture loading and unloading.

Figure 5.5 Web-based fixture motion planning system

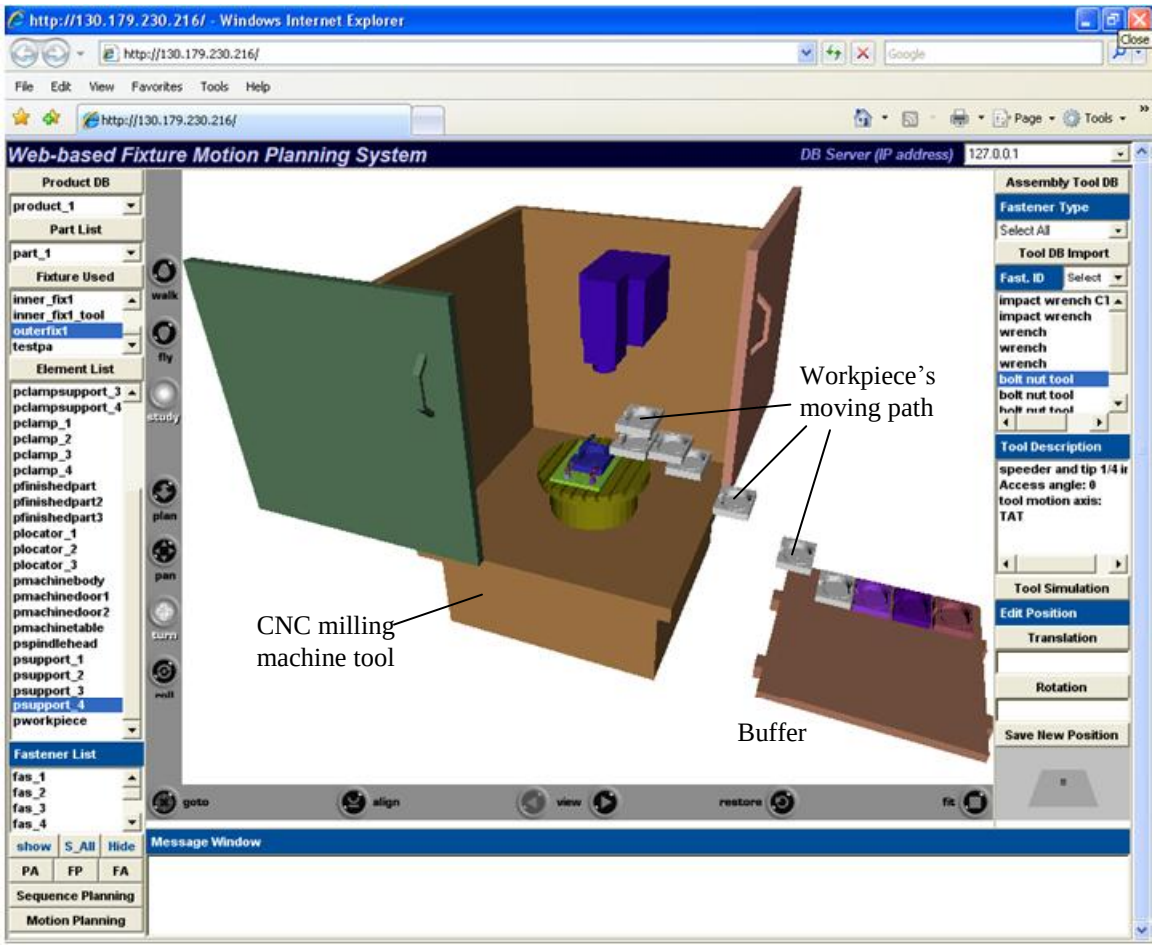


Figure 5.6 shows the generated motion plan in fixture loading and unloading, and the spherical accessibility map in the fixturing position of a workpiece. Each pixel in the spherical accessibility map represents a moving direction starting from the current position. The black area represents inaccessible moving directions, white area represents accessible directions. By clicking the function button “unloading simulation” and “loading simulation”, the unloading and loading simulation will start. Clicking the “Part accessibility matrix >>” button will show the accessibility map at different positions. Figure 5.7 shows the spherical accessibility map of the workpiece at different positions along the generated path. As the workpiece moves away from the fixture, less directions are blocked by obstacles, so the accessibility map has more white spaces.

Figure 5.6 The Generated motion plan and the accessibility map at the fixturing position

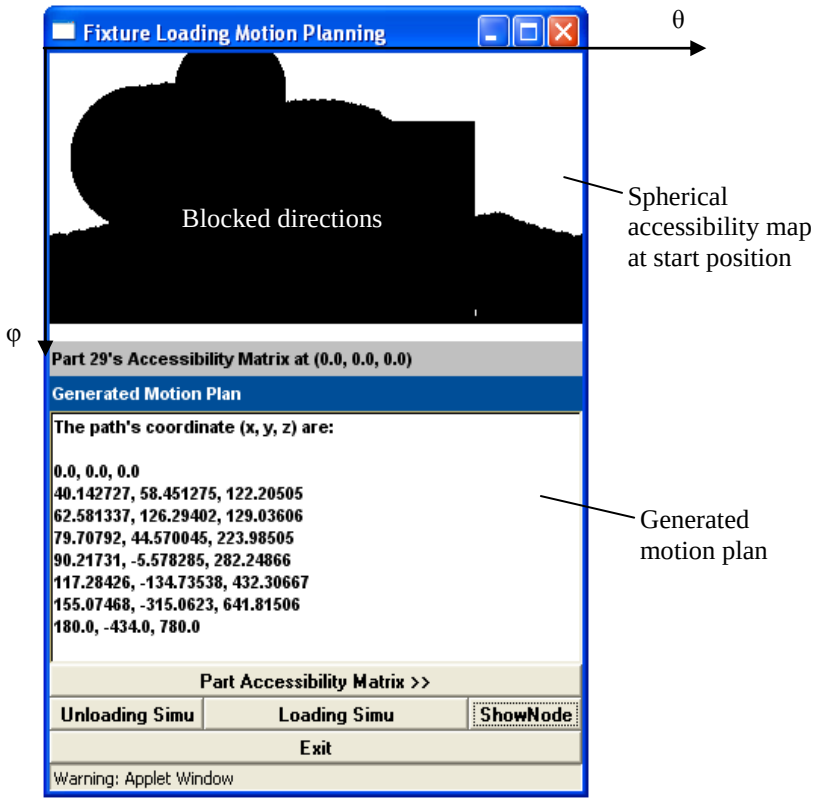
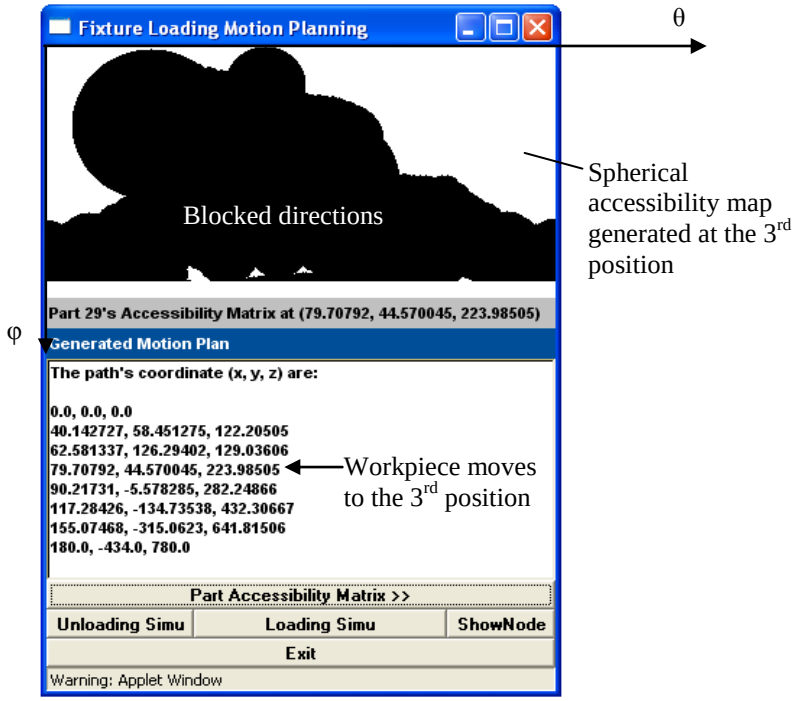
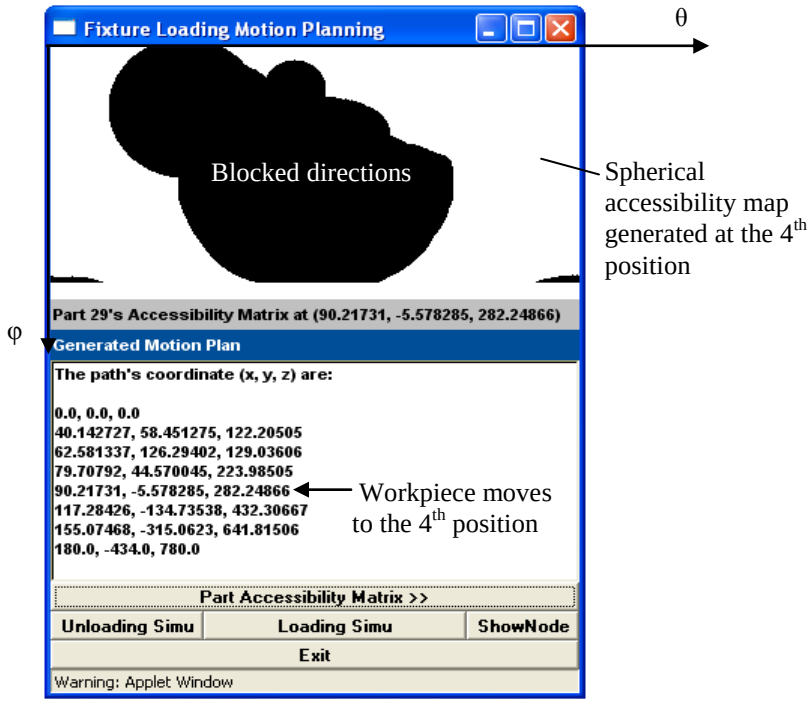


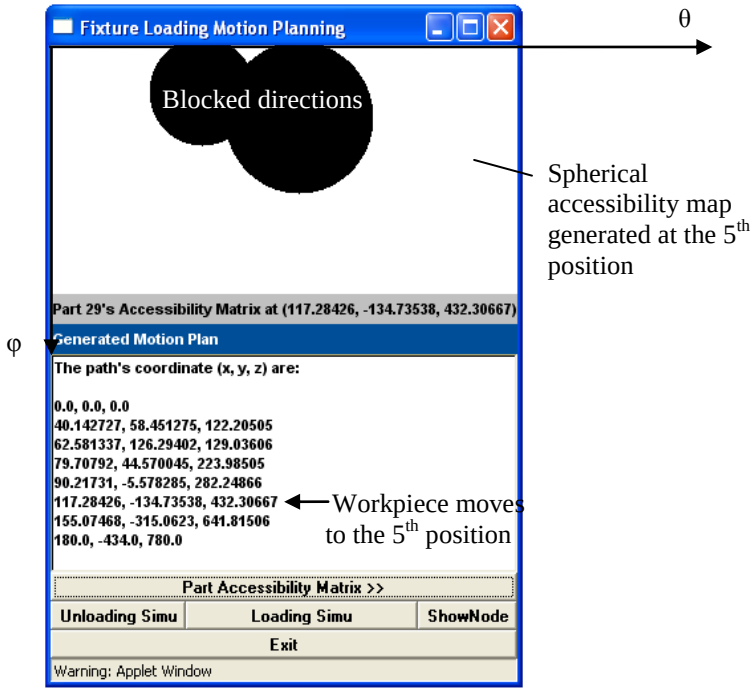
Figure 5.7 The accessibility maps of the workpiece at different position (a) At (79.71, 44.57, 223.99) (b) At (90.22,-5.58, 282.25) (c) At (117.28, -134.74, 432.31)



(a)



(b)



(c)

5.5 Chapter summary

This chapter presents a new accessibility-based A Star (A*) algorithm for motion planning in fixture loading and unloading. A spherical accessibility matrix is developed to identify accessible moving directions of a workpiece based on geometrical reasoning. A* algorithm is integrated with the spherical accessibility matrix to search for the shortest path. Random sampling is used to generate possible moving directions. A Web-based fixture motion planning system has been implemented. The implementation result of the proposed algorithm shows that feasible collision-free motion plans can be generated. The algorithm is efficient based on experimental tests. The proposed algorithm enables fast evaluation of fixture loading and dynamic simulation of the generated motion plan. It can be used in automated part loading and unloading systems.

Chapter 6

A Web-based virtual environment for fixture planning

This chapter presents the construction of a Web-based VE for fixture planning. The integration of CAD models with fixture planning via the VRML data format is discussed including the fixture assembly model extraction, assembly constraint recognition from CAD-VRML data, fixture visualization, and dynamic simulation of the generated assembly plan in a Web-based VE. A general translator of CAD-VRML data is implemented using Java techniques. MySQL database is used to store the extracted fixture information. The fixture assembly plan is simulated using VRML EAI. The Web-based fixture assembly planning system is implemented in a three-tier client/server architecture consisting of a Web server, a DB server and clients. HTML, Java applet and VRML browser are adopted for the user interface on the client side to simulate the fixture and assembly plans.

6.1 Data integration of CAD models and fixture planning

Fixtures are normally modelled and designed in a CAD system. The collaboration between a fixture designer using CAD systems and an assembly planner using Web-based fixture planning systems is beneficial to both sides. A fixture assembly plan is generated based on the fixture design. Sometimes there are difficulties in generating a feasible fixture assembly plan due to geometrical constraints. Such difficulties may be solved by the design modification. An assembly planner can suggest the design modifications according to the assembly evaluation result. Then a fixture designer revises the fixture for corresponding modifications. One of the critical issues in the above-mentioned process is the data integration of CAD models and fixture planning systems.

A fixture assembly model may be exported to some neutral data formats for the data exchanging purpose such as VRML, STEP, and DXF. Among them, VRML files directly support Web-based applications, which is a trend for the data share and reuse. VRML data can be compressed into a smaller size than most other 3D data formats, which can be transferred easily over the Internet. When 3D objects and scenes are described by the VRML format, animations of objects can be triggered by external events through the addition of program codes (e.g. Java or JavaScript). As an open standard file format, VRML is chosen for this research. A commercial CAD system, Pro/ENGINEER, is used to construct digital models of the fixture assembly.

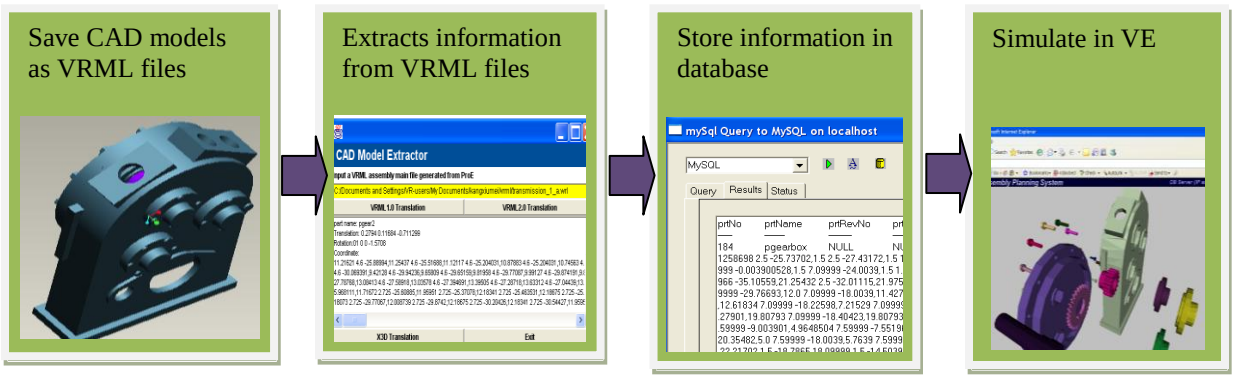
Figure 6.1 shows an overall system structure for the integration of CAD models with fixture planning. A fixture assembly consists of fixture elements and fasteners. Fixture ele-

ments are locators, clamps, supports, and other accessories. Fasteners mechanically join or affix two or more elements together, such as a screw. In a CAD system, the fixture assembly are exported as a set of VRML files, including a VRML master file and several VRML files. The VRML master file contains the file names and transformation information of fixture elements and fasteners. Each fixture element or fastener is described in a separate VRML file.

A VRML extractor is developed to extract useful fixture assembly information from VRML files. Fixture design data are divided into reference data and geometrical data. Reference data include fixture elements' name, fasteners' name. Geometrical data include the scale factor, transformations, coordinates, coordinate indexes, and so on.

The extracted VRML data are saved in DB Server and used in the fixture assembly planning system. Geometrical analyses include the fastener-element connectivity, element accessibility, and fastener accessibility. The feasibility analysis of assembly tools are integrated into the assembly planning. Finally, the VRML models are reconstructed in a Web-based VE, and the generated assembly plan is animated in the VE.

Figure 6.1 Data integration of CAD models and fixture planning



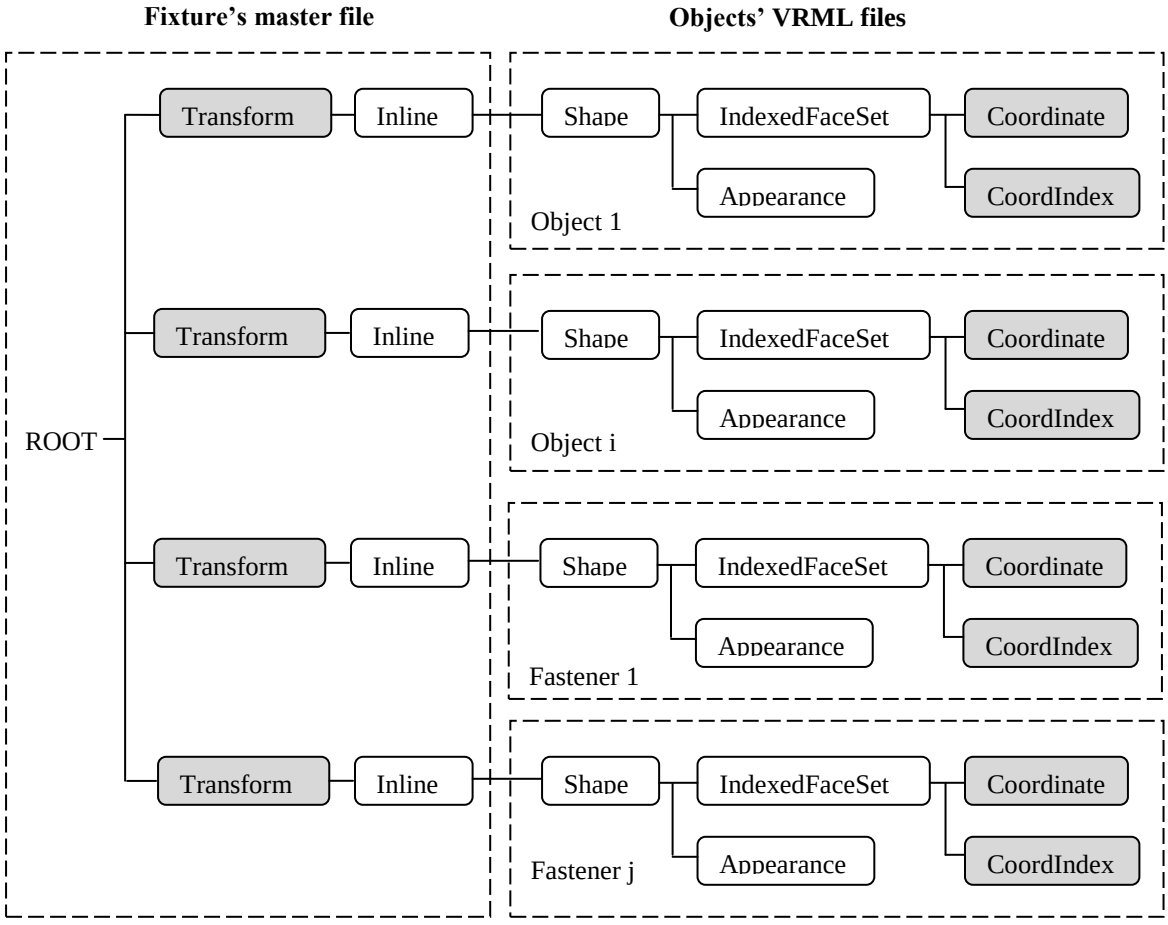
6.2 Extracting geometrical data from VRML files

The proposed system can generally use any CAD models from different CAD systems as most commercial CAD systems can export parts or assemblies into the VRML data format. However, different methods may be used to record VRML files in a specific CAD system. Since the reasoning algorithm is based on triangle patches of fixture models, the generated VRML files should use triangle patches for fixture geometry description. For the CAD systems using triangle patches in VRML files, the fixture models can be directly exported as ".wrl" (VRML) files, as those used by Pro/E. For those CAD systems using other geometry descriptions in VRML files, the fixture models should be first saved as a neutral file format (e.g., .dxf) then converted to VRML files in Pro/E, such as 3D Studio. The following description is based on the VRML files generated in Pro/E.

Separate VRML files are used for different elements in an assembly with an *Inline* function to link them. CAD models describe geometries with parameters, relations and references while the VRML format uses a set of triangle patches called *IndexedFaceSet* to record various faces. The directly generated VRML files contain a lot of extra nodes and data to describe the scene, such as viewpoints and groups. To reduce the data size, only useful geometry information is extracted from the VRML files. VRML data contain a set of objects such as 3D geometry, MIDI data, and JPEG images. These objects are called *nodes*. Each node consists of a type name, some *fields* and a set of associate *events* to describe its property and interactive actions. *Fields* define the initial parameters of a node such as dimensions, colors and positions. The value of *fields* can be changed via incoming *events*. Such change may trigger an outgoing *event*. Figure 6.2 illustrates a VRML

file structure of a fixture assembly. It includes an assembly's master file and a set of object files. The VRML master file contains all objects' file names and their transformation. An object's VRML file name is specified in the *url* field of an *Inline* node. It might be a fixture element or a fastener. Geometry and appearance data are recorded in an object's VRML files. The scene graph contained in all files forms the whole scene. The important node which contains extracted information is marked in grey.

Figure 6.2 The VRML file structure of fixture assembly



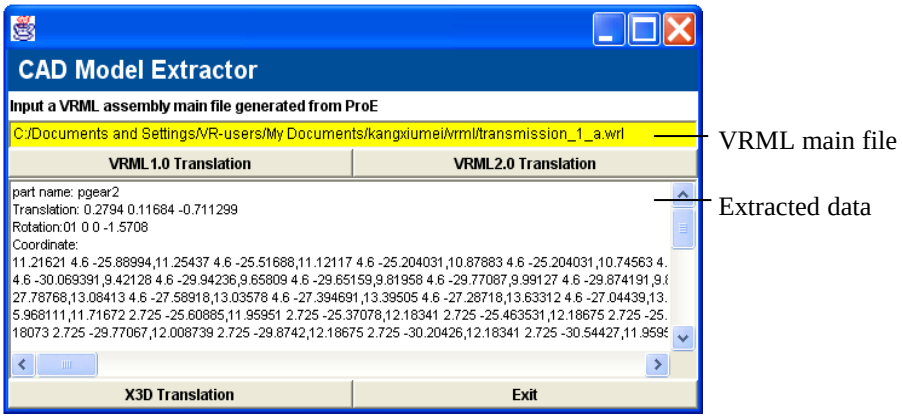
A 3D object in VRML is defined by its geometry, its appearance (color, lighting and texture) and its transformation (position, orientation, scaling). An object's geometry is placed in the fixture assembly with a specific position and orientation. Such adjustment is defined in a *Transform* node. VRML uses a Cartesian coordinate system. All geometry nodes are related to a local coordinate system and affected by parent transformations which accumulate down the scene graph hierarchy. A *Transform* node defines a coordinate system for its children. The transformation information including rotation, scale and translation values is extracted from the main assembly file. The rotation field specifies a rotation of the coordinate system about a rotational unit vector. The translation field defines a translation of the coordinate system. The scale field defines a non-uniform scale of the coordinate system about an arbitrary point. All 3D points are transformed into corresponding points in their parent's coordinate systems after a series of transformation.

An object's geometry is normally represented as *IndexedFaceSet* node under *Shape* node. *IndexedFaceSet* defines an object by a set of planner faces (e.g. triangle patches) in the local coordinate system. Fields of *IndexedFaceSet* include *coord*, *coordIndex*, *normal*, *normalIndex*, and so on. The *coord* and *coordIndex* are extracted from *IndexedFaceSet* as geometrical data. In *IndexedFaceSet*, the *coord* field specifies a *Coordinate* node which gives a set of 3D coordinates. The *coordIndex* field defines the faces by a set of coordinate indexes. For example, a triangle patch is defined by three point indexes. The marker -1 is used to separate indexes for faces. *Normal* field defines a *Normal* node which is specified by a set of vectors. *NormalIndex* is applied to *Normal* in the same manner with *coordIndex*. The *color* field defines a *Color* node to determine a list of colors for faces.

Since other fields only affect the appearance of the object and can be assigned in the program, only the basic geometrical information is extracted including coordinate, and coordinate index.

A VRML file extractor is developed using Java. The 3D geometrical information is extracted from VRML files by the extractor and then stored in the fixture database. Figure 6.3 shows the extracted data from VRML files. The extracted information including the fixture element's or fastener's name, its transformation, its coordinates, and its coordinate indexes is displayed in the text box.

Figure 6.3 VRML file extractor for CAD models



6.3 Use extracted data for fixture planning

In order to integrate CAD models with fixture planning, two important aspects are considered: the use of objects' geometry data in the fixture analysis, and the representation of objects in the virtual world to show the objects and to simulate the fixture planning results. Since the extracted data only preserve the geometrical information, assembly con-

straints between each pair of objects is lost. Geometrical reasoning according to the coordinates of triangle patches is required for automatic fixture planning.

6.3.1 Transform objects to the world coordinate system

After an object's coordinate and transformation information is extracted from VRML files, all objects in the assembly are transformed into a world coordinate system for fixture planning. The geometric 3D transformation consists of *translation*, *rotation*, *scale*, *scaleOrientation* and *center* fields. It may include a non-uniform scale about an arbitrary point, a rotation about an arbitrary rotational unit vector and a translation. These transforms are applied to the local coordinate system of each object to fit in the world coordinate system.

Assume the vertex $P(x_1, y_1, z_1)$ is a point in a object's coordinate system. P is transformed to $P'(x_1', y_1', z_1')$ in the world coordinate system by a series of transformations including rotation, scaling, and translation. In the matrix transformation notation, R, S, and T represent rotation, scale and translation, respectively. P' can be calculated using Equation (6.1) (Carey and Bell, 1997).

$$P' = P \cdot S \cdot R \cdot T \tag{6.1}$$

In VRML 1.0, the transformation is given in the form of transformation matrix including rotation and translation. In VRML 2.0, rotation is expressed as a normalized rotational unit vector (x, y, z) and an angle (α) about this unit vector in radians. It can be converted to a 3x3 rotation matrix (Baker, 2008). Here the matrix is transposed to be compatible

with a row expression of point *P*. Assume the translation is (x_0, y_0, z_0) and scale is a uniform scale at *s*. The *4x4* transformation matrix *M* can be converted as Equation (6.2).

$$M = s \cdot \begin{bmatrix} 1+(1-\cos\alpha)(x^2-1) & z\sin\alpha+(1-\cos\alpha)xy & -y\sin\alpha+(1-\cos\alpha)xz & 0 \\ -z\sin\alpha+(1-\cos\alpha)xy & 1+(1-\cos\alpha)(y^2-1) & x\sin\alpha+(1-\cos\alpha)yz & 0 \\ y\sin\alpha+(1-\cos\alpha)xz & -x\sin\alpha+(1-\cos\alpha)yz & 1+(1-\cos\alpha)(z^2-1) & 0 \\ x_0 & y_0 & z_0 & 1 \end{bmatrix} \tag{6.2}$$

6.3.2 Reconstruct VRML models for visualization in VE

In fixture planning, 3D geometry data of objects are represented by a Java class *VRMLGeometry*. It contains various properties and methods. The properties consist of VRML node name, object name, object type (a fixture element or fastener), basic fastener types, the total number of coordinates in the VRML node, the total number of coordinate indexes, coordinates of the VRML node, coordinate indexes of the triangle patches, and so on. These data are further used in the developed methods for fixture planning. The class *VRMLGeometry* is shown as follows:

```
public class VRMLGeometry {
    String sVRMLnodeName;
    String sObjectName;
    int iObjectType;
    int iBasicFastenerType;
    int iNumOfCoordinate;
    int iNumOfCoordinateIndex;
```

```
    float[][] fCoordinate;
    int[][] iCoordinateIndex;
    VRMLGeometry(){…}
    public String getVRML() {…}
    … …
}
```

The process *getVRML()* reconstructs the geometrical data of a object into VRML file format. The data of each object are defined as a *Shape* node. It uses an *IndexedFaceSet* node as the geometry field value. The data of coordinate and coordinate Index come from *fCoordinate* and *iCoordinateIndex* in the *VRMLGeometry* class. The generated *Shape* node is written as:

Shape {

geometry IndexedFaceSet {

solid FALSE

coord Coordinate {point [x_1 y_1 z_1, x_2 y_2 z_2, … …]}

coordIndex [i_1, i_2, i_3, -1, i_4, i_5, i_6 , -1, … …]

appearance Appearance {material Material{… …}}}

6.3.3 Geometry data representation in VE

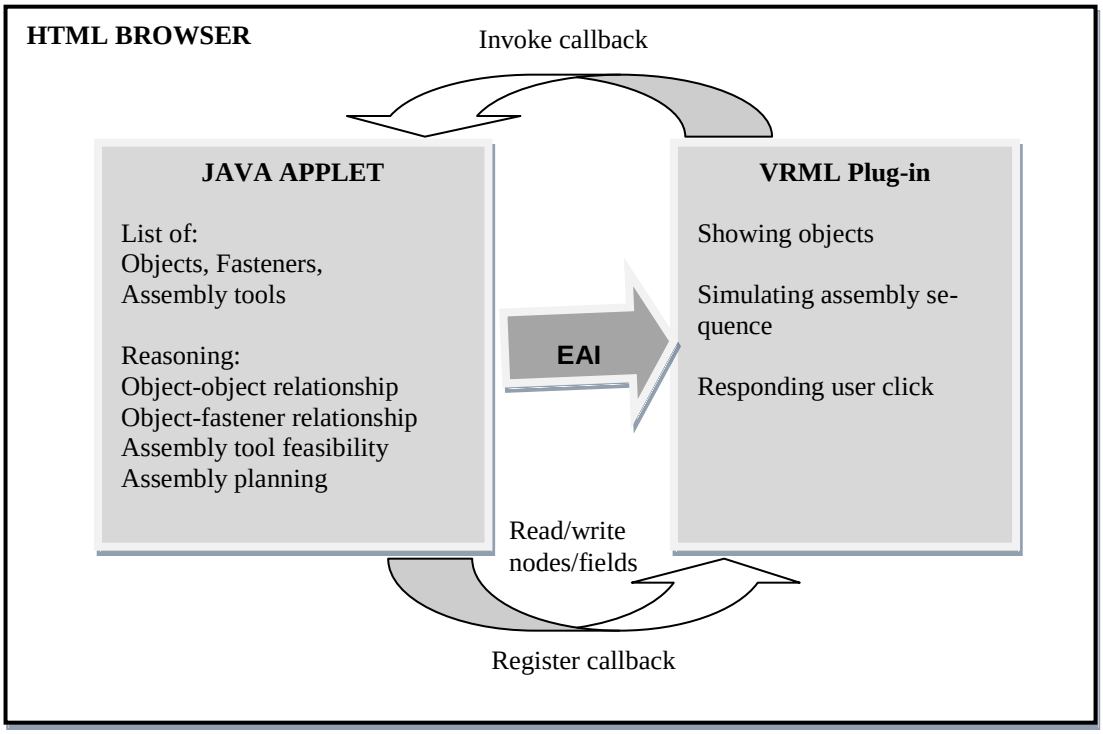
The user interface for fixture planning is displayed as a HTML Webpage with an embedded VRML browser plug-in and Java applets. 3D objects and the fixture plan are simulated via the VRML browser. The 3D objects can be accessed and manipulated by Java

applet, such as selecting an object, changing the color of the object, and moving the object to a specific position and orientation. Thus, various fixture assembly plans and moving paths are simulated visually to further validate analysis results. Figure 6.4 shows the communication between a Java applet and a VRML plug-in in a Web-based application. The communication between the Java applet and VRML browser is accomplished by VRML EAI. The EAI is a programming interface enabling bilateral communication between VRML and external programs such as Java applets. It allows Java applet to read and change the scene graph, and invoke a registered callback to an event occurred in the VRML world (Diehl, 2001).

The 3D geometry and fixture assembly plan is simulated in the VE using the geometry data. *DEF* and *USE* are two keywords in VRML to enable nodes reuse. *DEF* defines an object's scene graph with a specific name. The object's scene graph can be inserted at different positions in the scene graph by calling the defined name. *DEF* is used to assign a node name to an object with its geometry data. A fixture element or a fastener is assigned a number after its name is sorted. The element is defined as $part_i$, i=1, 2……n. and the fasteners as $fastener_j$, j=1, 2, ……m. The node names of these elements and fasteners are predefined as empty *Transform* nodes in a VRML file (e.g. originalScene.wrl). In the implementation, this file is embedded in HTML file and initially loaded in the VRML browser. When a user clicks the simulation button, the Java applet requests a handle of the VRML plug-in by calling up the BrowserFactory.getBrowser() method of the Browser class. Then the predefined object or fastener node is retrieved by using getNode() method of this handle. The createVrmlFromString() method of the class is applied

to set values to element's or fastener's nodes using their geometry data. Thus geometry data from Java applet are represented as nodes in the VRML browser.

Figure 6.4 The communication between VRML plug-in and Java applets



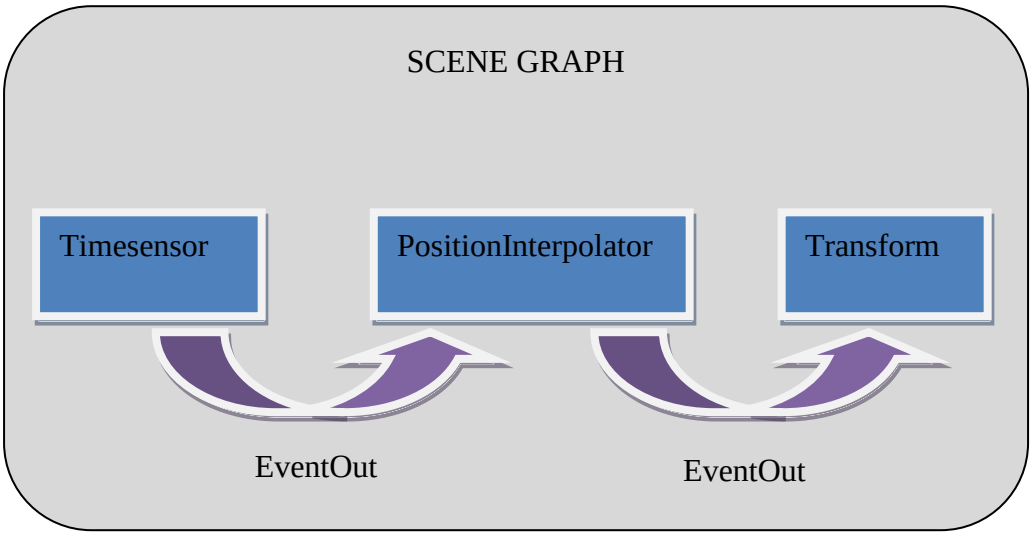
6.3.4 Dynamic animation in VE via VRML EAI

There are several techniques in VRML to generate interactive animations, mainly routes, scripting and EAI. A node uses events to receive signals and respond to other nodes. *Events* consist of a number of incoming events (*EventIn*) and sent out events (*EventOut*). *EventIn* changes node's status while *EventOut* send out a signal to other nodes to indicate such change in the node.

A geometry bounded to a node with *DEF* can be dynamically manipulated by external programs. VRML scenes react to a variety of external events by using several *Sensors*. *Sensors* generate *eventOut* events as the reaction to external events such as user interaction or passing of time. For instance, a *TouchSensor* node is attached to an object. When a user touches an object in VRML, the *TouchSensor* generates *eventOut* event and changes an event variable in VRML. This event variable of the VRML node is assigned to a *TouchSensor_changed* variable in Java applet. Then a *VRMLEventListener* is added to this *TouchSensor_changed* variable to await event message. Thus events that occur in VRML world are linked to Java applet.

EventOuts send events to other nodes via *routes*. In turn, *eventIns* receive the value of a field as an event via routes. An *Interpolator* defines a linear function given by key values. The value between key values is linearly interpolated. An *Interpolator* receives events from *Sensors* and sends out value to other nodes. A *PositionInterpolator* is one of the Interpolators to distribute positions linearly. It can be used to animate the motion of an object. Figure 6.5 illustrates the animation concept of objects through *TimeSensor*, *PositionInterpolator*, and *Transform*. A *TimeSensor* begins to generate events directly after it is created. This eventOut *fraction_changed* is of SFFloat type and the value is in the interval [0...1]. The eventOut *fraction_changed* is routed to eventIn *set_fraction* of a *PositionInterpolator* node. This node generates eventOut *value_changed* as type SFVec3f which gives 3D coordinate values. The values are further sent to the eventIn *set_translation* of the *Transform* node. So the object moves to a new position accordingly. The position of an object can be changed gradually based on the generated motion plan.

Figure 6.5 Motion animation in VRML scene graph

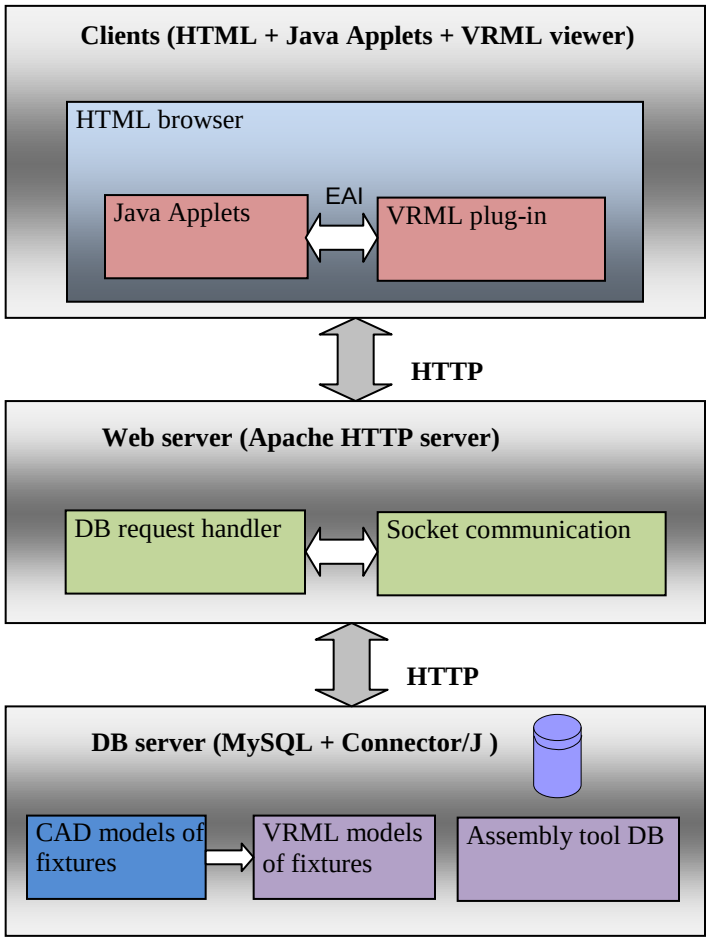


6.4 Implementation of the Web-based Virtual Environment

A three-tier client/server architecture is implemented including a Web server, a DB server and several clients. A client is a program that sends operational request to a Web server. The Web server accepts the request, performs the requested service, and return results to the client. A separate DB server is used to communicate with the Web server and to provide the data-related service for the Web server. The advantages of the three-tier client/server architecture are the performance improvement for groups with a large number of users and the flexibility compared to the two-tier approach. Such thick client architecture can reduce the computational intensive analysis of the Server. It doesn't require a

Server with high level of performance. Figure 6.6 shows the three-tier client/server architecture used in the developed fixture planning system.

Figure 6.6 Three-tier client/server architecture for WFAPS



On the client side, Java applets and VRML browsers are embedded in a HTML Web page. Cortona client is used as the VRML viewer plug-in. The 3D models of a fixture assembly and assembly tools are retrieved from a user-designated remote DB through the Web server. The fixture plan is simulated by the interaction with 3D objects in the

VRML world via VRML EAI. The transform matrix of each fixture element is stored in DB so that only one set of VRML geometry data is required for each fixture element with different orientations.

The Apache HTTP server runs as a middle tier. It handles request of the clients to DB server with socket-based communication. The HTTP service is bound to a specific port for HTTP request. It accepts queuing from clients for retrieving the geometry data from DB server and returns results to the clients. Socket-based communication is used to transfer geometry data and information between DB server and clients via a developed server program named "DataServer" which is bounded to a port and running as a service. The "DataServer" service listens to the port and waits for clients to make a request. The DB handler performs the request and retrieves data from DB server, as well as stores information to DB server.

DB server serves as the third tier and MySQL is adopted as the DB server. As the most popular open source for the structured query language (SQL) database management system, MySQL database works in client/server or embedded systems (MySQL AB., 2007). The MySQL Java Database Connectivity (JDBC) is a Java API for the connectivity between the Java programming language and MySQL database. It allows both local and remote database access. MySQL Connector/J is used as a native Java Driver that converts JDBC calls into the networked protocol used by MySQL. The CAD models of a fixture assembly are firstly constructed in CAD software on the server side and exported as the VRML file format. Then the assembly position and orientation of the workpiece, fixture elements, and fasteners are extracted from VRML files and stored in the DB server via a

local user interface. These data are retrieved by a client over the Internet through remote JDBC connection.

Assume that a modular fixture called "*modularFixture*" consists of several elements and fasteners. The CAD assembly models are constructed in Pro/ENGINEER and exported to a set of VRML files. The master file is "*modularFixture_asm.wrl*". It contains names of the fixture, the scale factor, names of all fixture elements and fasteners, and their corresponding transformations (rotation and translation). The following is the information contained in the fixture master file:

```
#VRML V2.0 utf8
……
DEF V_Model Anchor {
  url ""
  description "modularFixture.asm"
}
… …
s_factor 39.3701
… …
Transform {
rotation 0 -4.49808e-16 1 3.14159
translation 5.74548 0.1778 3.4145
children [
Inline {
```

url "locator_1_prt.wrl"}

… …

Transform {

translation 4.72948 0.73152 3.9225

children [

Inline {

url "fas_1_bolt_prt.wrl"}

In the above file, an object named “locator_1_prt” is found via the *url* of an *Inline* node. The VRML file name “locator_1_prt.*wrl*” is automatically searched. The geometrical data of locator_1 are extracted from the *IndexedFaceSet* under *Shape* node. The information in “*locator_1_prt.wrl*” is shown as follows:

#VRML V2.0 utf8

... ...

Shape {

geometry IndexedFaceSet {

… …

coord DEF FaceC Coordinate {

point [

0.09864598 0 -0.024318214,

0.089962736 0 -0.047213012,

0.076050394 0 -0.06737096,

0.057711086 0 -0.083617308,

... ...]}

coordIndex [

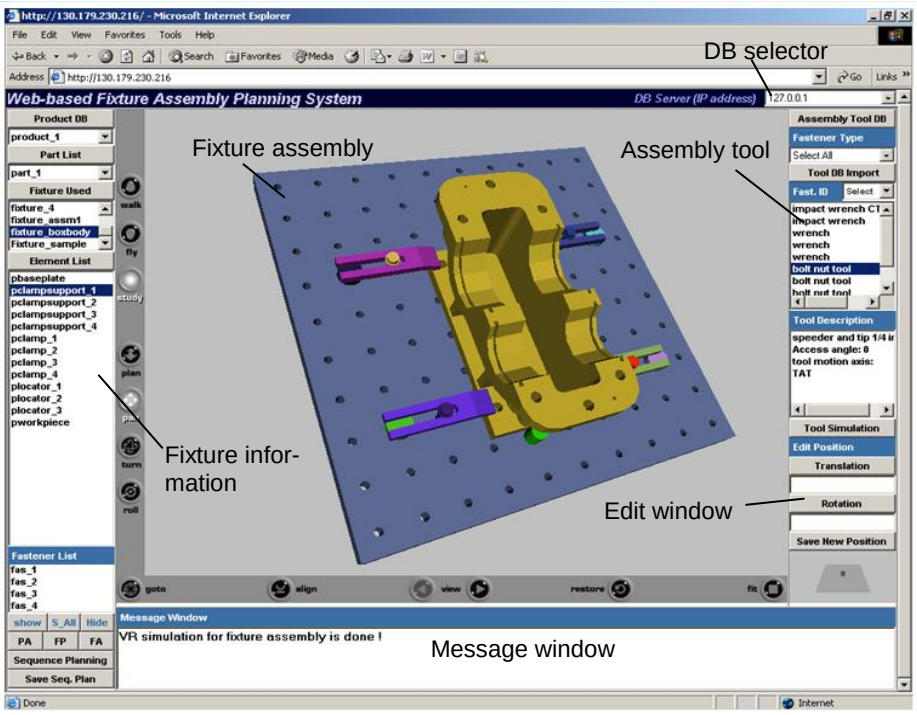
45, 44, 46, -1, 47, 46, 44, -1,44, 43, 47, -1,

48, 47, 43, -1, 43, 42, 48, -1,49, 48, 42, -1,

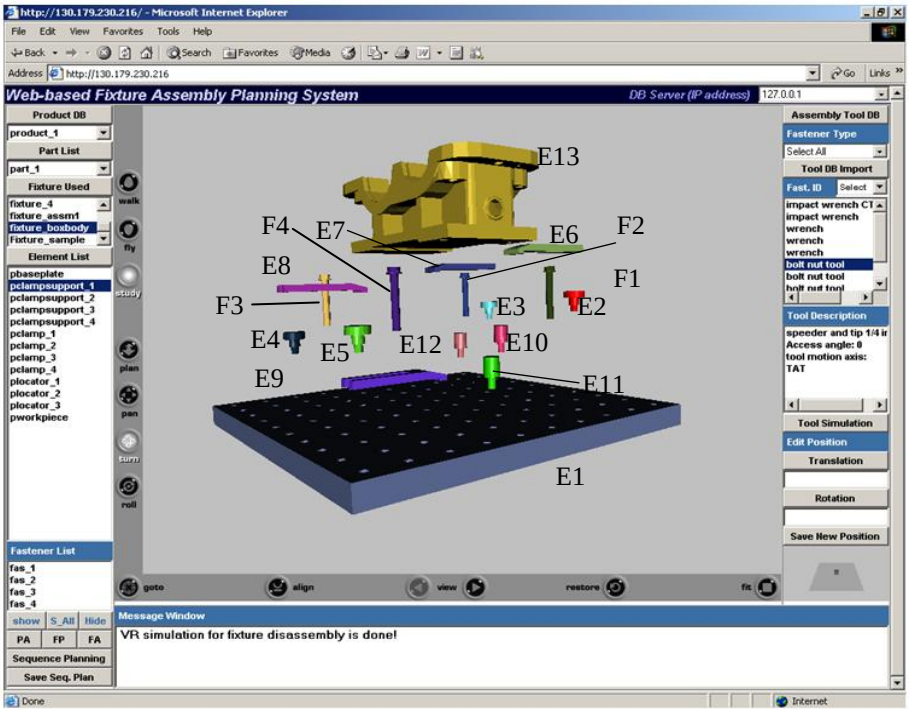
……]}}

Figure 6.7 shows the Web-based 3D visualization environment for fixture assembly planning on the client side. In Figure 6.7 (a), two Java applets locate in the left and right side of the HTML window respectively. The product, its subassembly, the fixture, fixture elements' list, and fasteners' list are shown in the left side Java applet. The information of assembly tools is shown in the right side Java applet. The VRML browser is embedded in the middle of the window showing the 3D fixture assembly model. First, a user needs to click the "Product DB" button so that a list of products is retrieved from the chosen DB, from which the user can choose one product. Then the subassemblies of this product are retrieved from DB. After selecting a subassembly and a fixture of this subassembly, the user can get all the fixture elements and fasteners by clicking "Element List" button. All models are shown in the VRML browser in the middle of the window. Similarly, the assembly tools are searched from the tool DB. Suitable tools are automatically chosen for a specific task. Figure 6.7 (b) illustrates the animation of a fixture disassembly process.

Figure 6.7 Web-based VE for fixture planning (a) the user interface for fixture assembly planning (b) simulation of the fixture assembly/disassembly process



(a)



(b)

6.5 Chapter summary

This chapter presents the construction of a Web-based VE for fixture planning. An integrated approach is discussed for 3D CAD models of fixture with a fully automated assembly planning system. The data extraction of geometric information from 3D CAD-VRML files enables automatic data integration without the human interaction for the fixture data input. The regroup and applications of fixture geometrical information is described. Dynamic simulation of fixture plan is realized via VRML EAI since Java applets can communicate with 3D models and manipulate the models in the VRML scene. Three-tier client/server architecture is implemented for the Web-based VE. It consists of a Web server, a DB server, and several clients.

The developed Web-based VE provides a means of conducting and visualizing fixture planning for both fixture designers and other engineers over the Web. It supports multiple clients to conduct fixture planning task from different computers and locations interactively and collaboratively. The data integration method improves the efficiency and visual effect of fixture planning. Errors and costs for data exchange are reduced.

Chapter 7

Construction and performance improvement for large-scale VEs

The VEs' performance decides the effectiveness of the fixture design and manufacturing using virtual reality. A large VE is extremely computational demanding for real-time rendering and manipulating. Cell segmentation is a method to partition the virtual world into smaller universes or cells. It is frequently used for rendering large models to minimize the impact of computer memory swaps on simulation frame rate. This chapter discusses an improved method in the construction of VEs using Eon Studio. It is an improved cell segmentation method with dynamic loading strategies for performance improvement of large-scale VEs. Two case studies are conducted using the proposed methods in building navigation and product assembly simulation. A large building model is used to show the system navigation performance improved significantly in the first case, comparing to the less improvement for a relative small product assembly model in the second case. Potentially visible sets (PVS) and regions are formed based on the attributes

of cells in VEs. The PVS and regions are then converted into prototypes. A dynamic loading strategy is applied to load these prototypes in the real-time simulation. The VE performance is improved when compared to the non-segmented model. In this chapter, Section 7.1 describes the construction of VEs for the model visualization. Section 7.2 presents the cell segmentation and dynamic loading strategies for performance improvement in a building navigation. Section 7.3 demonstrates the cell segmentation application in dynamic loading of a product assembly simulation.

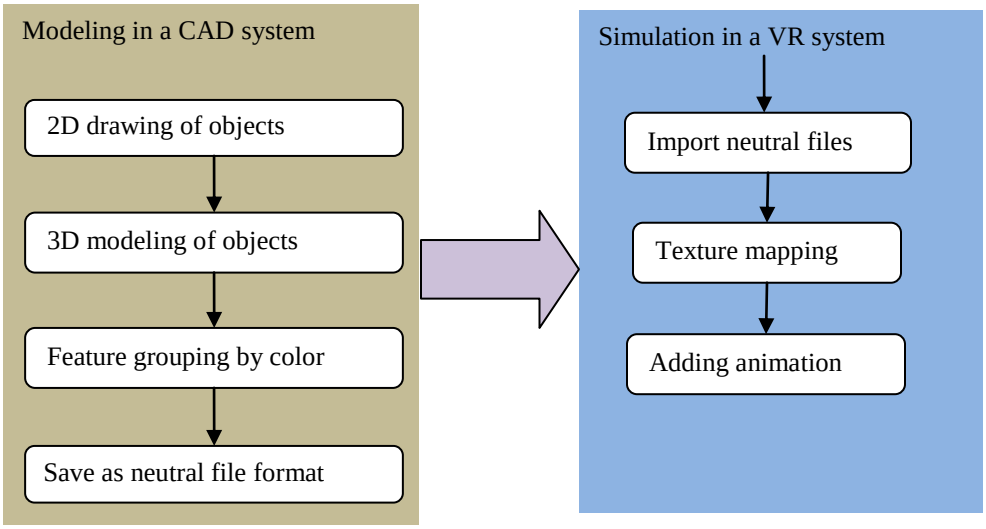
7.1 Construction of VEs

A VE consists of numerous 3D models with different colors, textures, properties and behaviours. The construction of a VE consists of two stages: products modeling in CAD systems, and the products simulation in a VR system. An overall workflow in the VE development is shown in Figure 7.1. In the first stage, it is necessary to convert any 2D drawings of products or architectures into 3D models, which is usually done in CAD software. The generated 3D models are then transferred into a VR system via a neutral data format, such as 3D studio's *.3ds. In the second stage, the VE is formed by object texture mapping, adding animation, adding light, etc. Eon Studio 6.1 is used as a VR simulation tool in this research.

The 3D modelling using CAD systems may take a lot of time to complete. After the simulation is generated in VR, modelling errors, missing objects and features are often identified in the simulation. It is common to modify a CAD model in CAD systems and import the improved CAD model to the VR system again. Reworking of texture mapping

and object rendering in a VR system is usually tedious and time-consuming. A rapid automatic texture mapping method is developed in this research to ease the workload. Following description is based on modeling of the virtual Engineering and Information Technology Complex (EITC) building at the University of Manitoba, a large-scale VE.

Figure 7.1 Overall workflow in the VE development



7.1.1 Modeling in CAD software

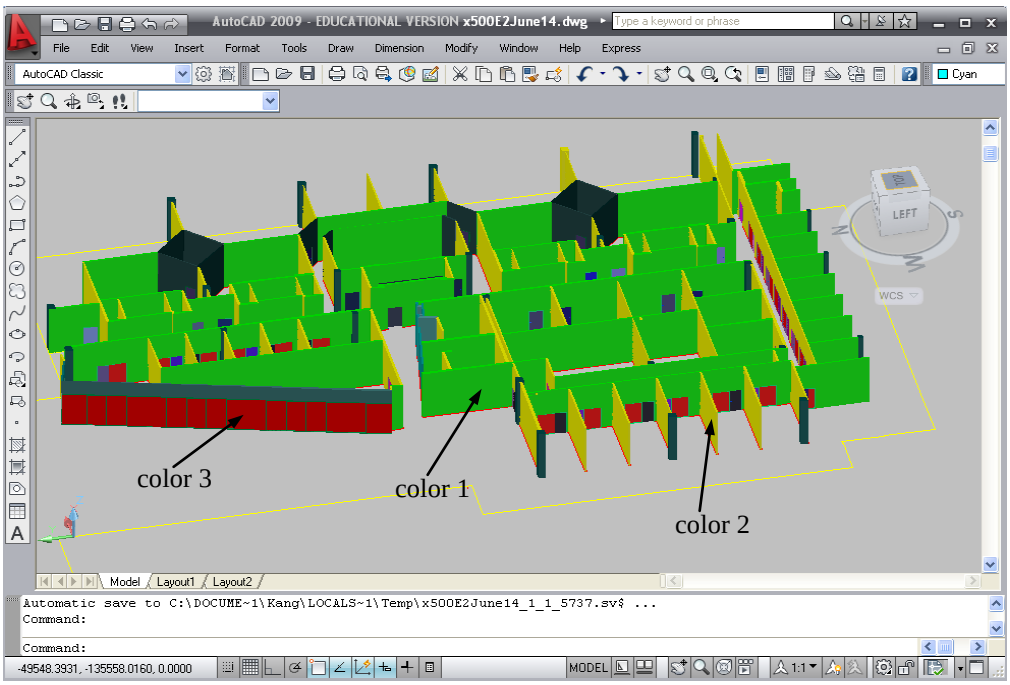
Numerous CAD packages support 3D modeling of objects, such as AutoCAD, Pro/Engineer, SolidWorks, Autodesk 3ds Max. For this segment of the research, Auto-CAD is chosen as the 3D modeling software. Based on the floor plan of the design drawing, walls, doors, windows, and furniture are identified and modeled in AutoCAD.

Features are grouped according to their texture, orientation, and animation. Color is used as the identity of features. When features with the same color are exported to EON, the VR system, they are jointly converted to one object named by its color. These features

are then rendered by the same texture and have the same appearance and animation in EON. Moreover, the orientation plays a role in texture mapping. Features with the same texture but different orientation requires to be separated for proper texture mapping in EON. Finally, features with different animation are assigned different colors. For example, doors are assigned different colors to separate them in the simulation. In order to reduce time for the texture mapping in EON, features with the same texture, orientation, and animation are grouped together. If one of the features is different from others, this feature is assigned a different color to distinguish it from other features.

Figure 7.2 shows a model of the 5th floor in Building E2 in AutoCAD 2009. Walls facing north (marked by color 1) and walls facing east (marked by color 2) are separated as different colors due to different orientations. Window glasses are separated by color 3. The doors are separated by other different colors due to different animation.

Figure 7.2 The generated model of the E2 5th floor in EITC building

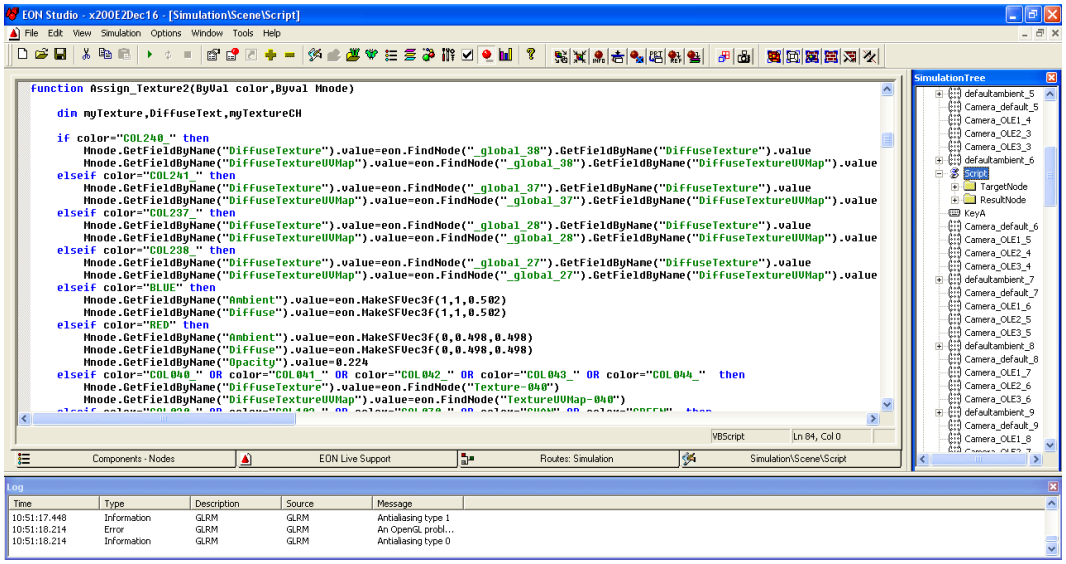


7.1.2 Simulation in VR

To import 3D models into the VR system, 3D objects generated in other data formats must be converted into a format that is compatible with EON Studio. Many formats are supported in EON including 3D Studio (*.3ds), DWG (*.dwg), Pro/Engineer .SLP (*.slp), VRML 2.0 (*.wrl)/VRML 97(*.wrl), to name a few. 3D Studio (*.3ds) is used to convert 3D models into EON.

After 3D models are imported into EON, the object's appearance in EON can be modified by changing its material and texture. A texture node can be applied to a mesh by placing a *Texture2* node beneath a Mesh node. A *TextureUVMap* defines the normal direction of a texture applied.

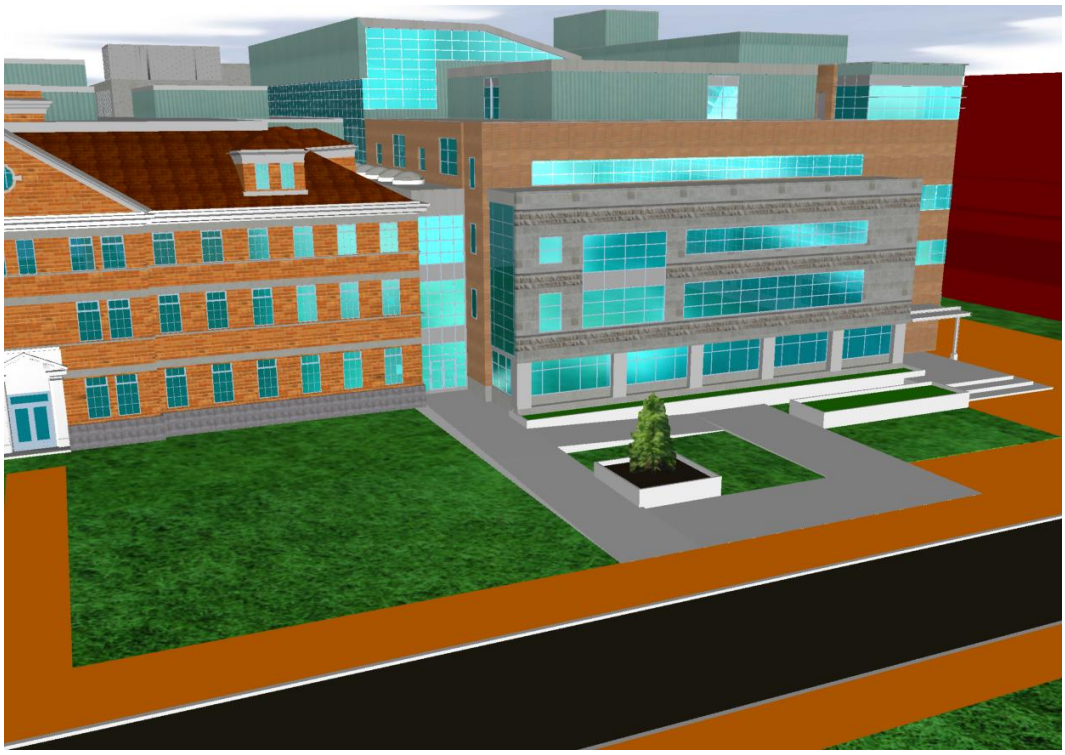
Figure 1.3 VB script for automatic texture mapping in EON



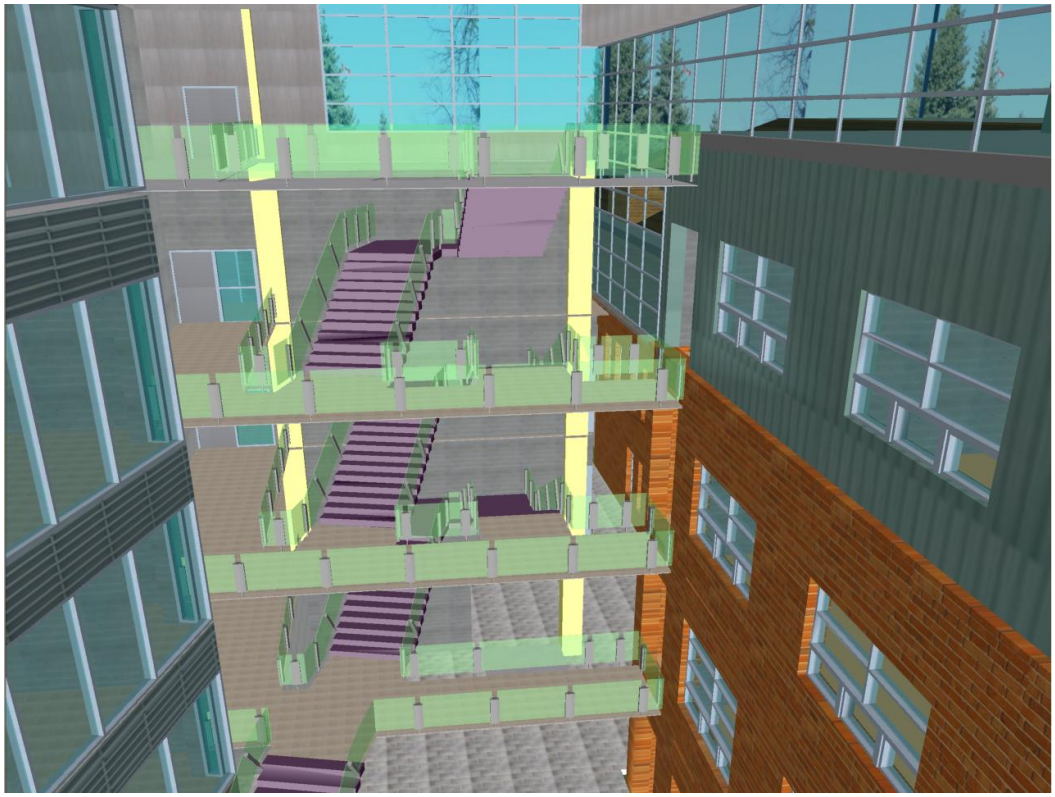
An automatic texture mapping using VB Script is developed in this research. Based on color grouping in CAD modeling, an object is identified automatically by using its color

name. Then predetermined nodes of *Texture*2, *TextureUVMap*, and *Material* are automatically applied to the object. Figure 7.3 shows the VB script for automatic texture mapping based on its color defined in CAD software.

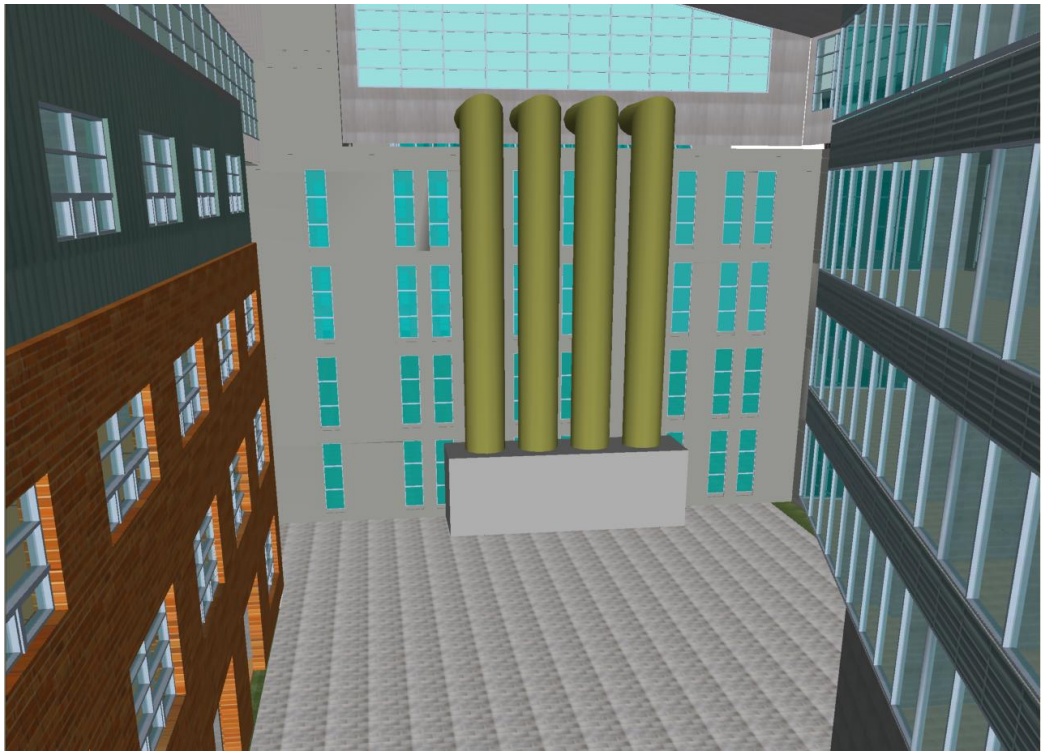
Figure 1.4 The generated large-scale VE of EITC building at the University of Manitoba (a) front view of EITC building (b) eastern view of the main atrium (c) western view of the main atrium (d) view of atrium in E1 building (e) view of a hallway at the 5th floor of E2 (f) view of a hallway in the 3rd floor in E1 building



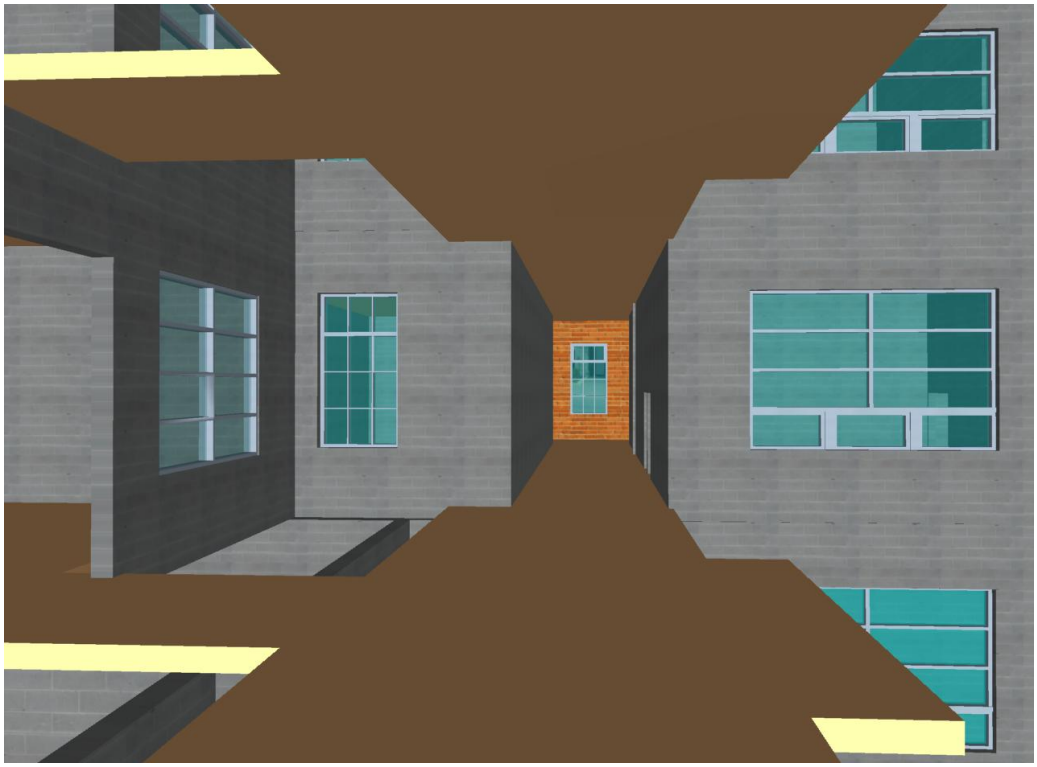
(a)



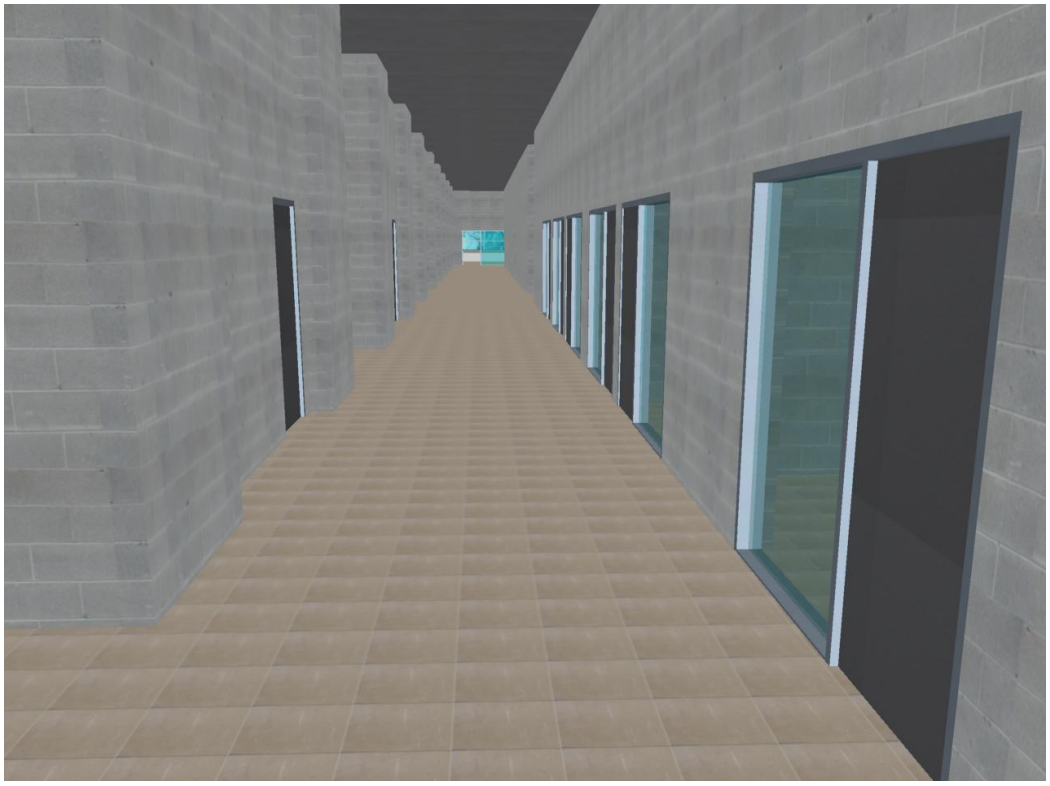
(b)



(c)



(d)



(e)

(f)

Different animations can be added to objects, such as opening a door and automatic navigation based on predetermined routes. For opening a door, a sensor is attached to the door. When a user touches the door, the sensor sends a signal to a *Place* node. The *Place* node changes the orientation of the door. The door is opened in the simulation. The generated large-scale VE of EITC building at the University of Manitoba is shown in Figure 7.4. Figure 7.4 (a) is the front view of EITC building. Figure 7.4 (b) and (c) display the main atrium of EITC building. Figure (d) is the atrium in E1 building. Figure 7.4 (e) and (f) shows two hallways in E1 and E2.

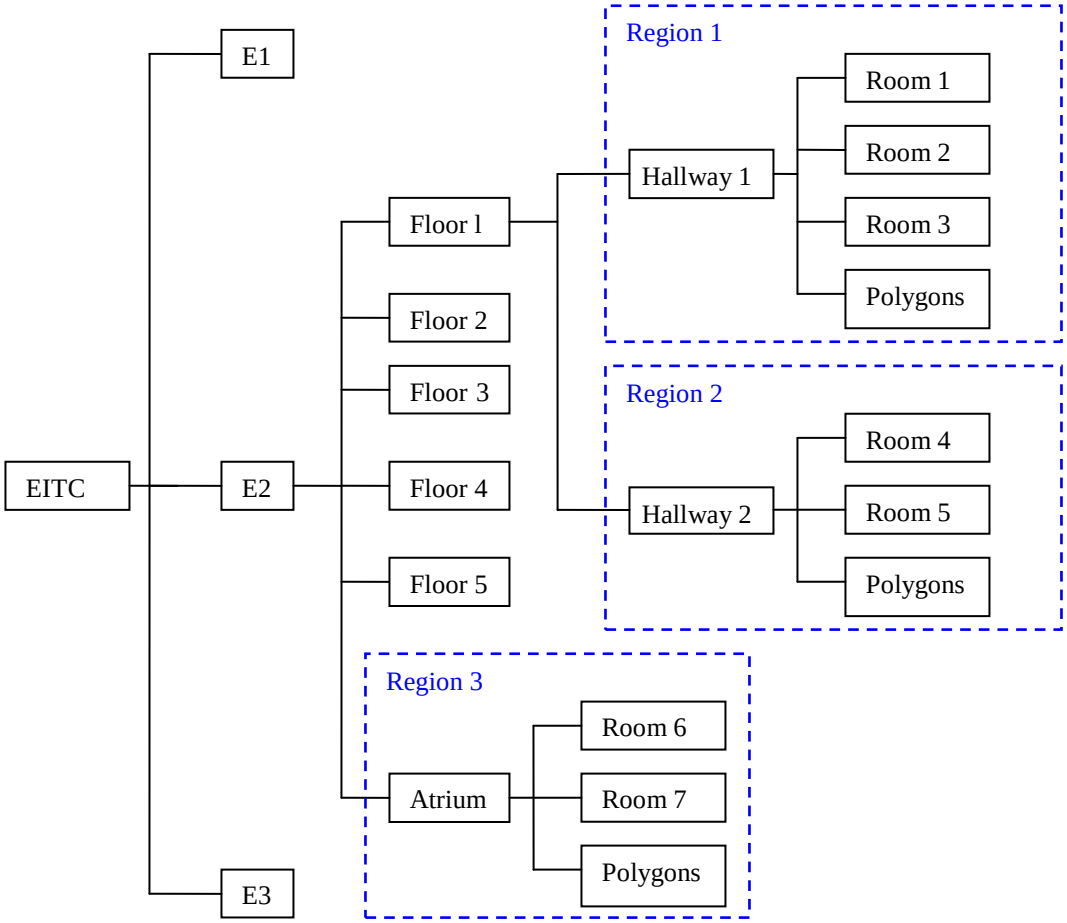
7.2 Case study 1: Dynamic loading in a large building VE

7.2.1 Overview of dynamic loading

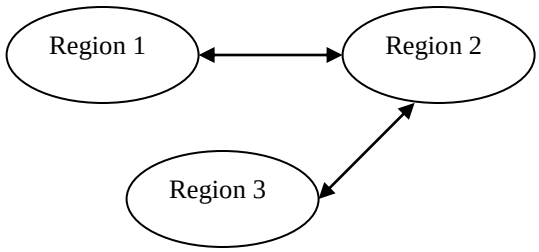
A building model can be divided into several relative regions, such as hallways, atrium and stairs. The PVS of these regions may include rooms with door(s) or window(s) and some potentially visible polygons through the portals. In a VE, a navigator can only see a small area of the whole model. Since other objects of the model are invisible to the navigator, it is possible to only load the current visible object to lessen the computer memory usage. For example, when a navigator walks through a hallway, he/she may look around outside or inside the room within the hallway, but it is impossible for him/her to see the room in the other hallway provided that there is no portal between them. If the navigator moves from one hallway to another, the system will load the next hallway and unload the previous one.

Figure 7.5 shows a scheme of the region dividing based on the EITC building models at the University of Manitoba. In Figure 7.5 (a), the EITC building consists of three sub-buildings E1, E2, E3. Each building has several floors with many hallways and rooms. The region is formed based on the hallways. Figure 7.5 (b) shows the adjacency graph representation of three regions. The adjacency graph is determined by the entry of a region. If there is an entry between two regions, these two regions are bilaterally connected. The adjacency graph is stored in a database. Each region is created as a prototype in EON and is stored in the prototype library of the system.

Figure 1.5 (a) A scheme of the region dividing of EITC building (b) regions' adjacency graph representation



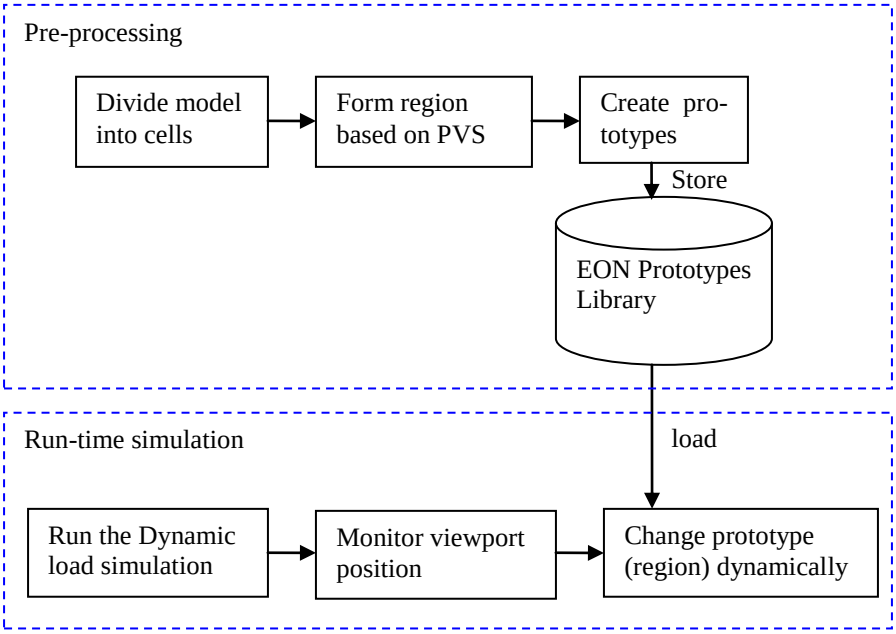
(a)



(b)

Figure 7.6 shows the architecture of the system. In the pre-processing stage, cells are grouped together based on PVS, and prototypes are created and stored in EON prototype library. During the run-time simulation, a navigator's position is monitored simultaneously. When an entry of a new region is detected, the corresponding name of the region will be searched from the database, and the prototype will be dynamically loaded from the EON prototype library.

Figure 1.6 The architecture of the dynamic loading system



7.2.2 Cell segmentation

Partitioning a building model into cells and portals is typically done by manually pre-processing when creating the model. For example, a room is created using a 3D modeling tool, the room consists of four walls. Some walls may have windows or doors. A room is

defined as the basic cell. A hallway consists of several walls of neighbouring rooms. Such cells are defined as same level cells.

Besides the same floor features, such as hallways and rooms, there are cross floor features, including atrium, stairs, elevators and exterior walls. The cross floor features consist of walls in different levels. For example, an atrium consists of walls in different floor levels. Such features are defined as cross level cells. The region dividing is based on the PVS of same level cells and cross level cells, which groups the adjacent visible cells together to form prototypes.

7.2.3 Forming regions based on PVS

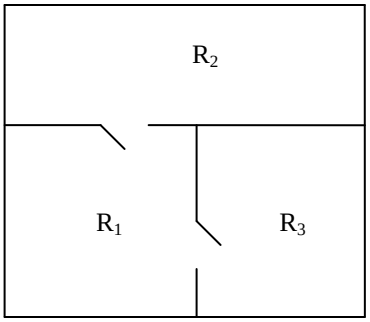
Let us examine the visibility of a room besides a hallway as shown in Figure 7.7 (b). If a room has a window or a door facing the hallway, that is, there is a portal in the cell, this room is visible to the viewer when a navigator walks through the hallway, such as room R_1. Furthermore, if this room has a neighbour and there is a door or window between them, the next room is visible to the viewer and should be included in the PVS, such as room R_7.

On the other hand, if there is no window or door facing the hallway in the room, for example, room R_2 in Figure 7.7(b). That means no portal faces the hallway. When a navigator walks through the hallway, this room is invisible to the viewer. This cell should be omitted from the region.

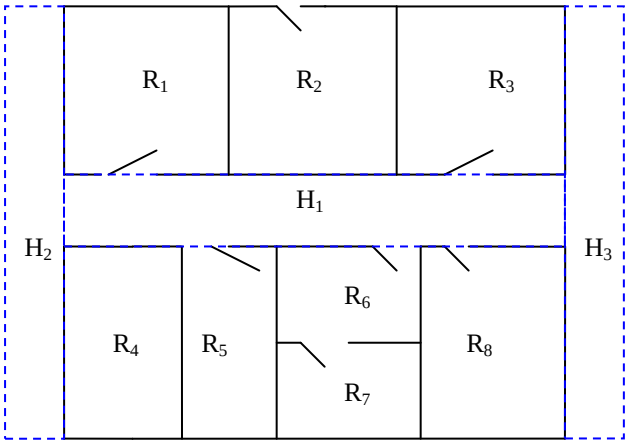
Figure 7.7 (a) shows the room R_1's PVS group G_{R1} forming based on R_1. Taking a room R_1 as a feature, an opening matrix $O_{R1}(n,w,s,e)$ is defined as the opening property of the

room in the north, west, south and east direction, respectively. If there is an opening in the direction, the value is 1, otherwise, the value is 0. A neighbour matrix $N_{R1}(R_n,R_w,R_s,R_e)$ defines the neighbours of this room in the north, west, south and east side, respectively. For example, the value R_n is the room number connected to room R_1 in the north direction so R_n equal to R_2. Then the group G_{R1} of room R_1 is expressed as:

Figure 1.7 Forming PVS (a) The R_1's PVS group forming (b) a hallway's PVS group forming



(a)



(b)

$$G_{R1}=O_{R1}*N_{R1}^{T}=(1\ 0\ 0\ 1)*(R_2\ 0\ 0\ R_3)^{T}=R_2+R_3 \quad (7.1)$$

Since the room R_2 is included in the group, its neighbours which have an opening to it may be visible if the navigator stays in R_1. Thus the group G_{R2} has to be calculated and included in the group G_{R1}. This calculation is recursively used until no further cell is included in the group G_{R1}.

Figure 7.7 (b) shows a hallway H_1's PVS group forming. It is a rectangle which only has neighbours in the northern and southern sides. To the contrary of a room's PVS forming, hallway H_1's PVS depends on its neighbours' property. If a room opens to a hallway, it will be included in the hallway's PVS group. The rooms R_1, R_2, R_3 in the northern side of H_1 have opening matrixes $O_1(0,0,1,0)$, $O_2(1,0,0,0)$, $O_3(0,0,1,0)$. The opening property of R_1 facing H_1 can be expressed as $O_1(1,3)$=1, which is the value at the first row, third column of matrix $O_1(0,0,1,0)$. If this value equals to 1, this room group should be included in the PVS group of H_1. The H_1 northern PVS group can be calculated as:

$$G_{H1n}=O_{H1n}*N_{H1n}^{T}=(O_1(1,3)\ O_2(1,3)\ O_3(1,3))(G_{R1}\ G_{R2}\ G_{R3})^{T}=(1,0,1)\ (G_{R1}\ G_{R2}\ G_{R3})^{T}=G_{R1}+G_{R3} \quad (7.2)$$

The H_1 southern PVS group can be calculated as:

$$\begin{aligned} G_{H1s}&=O_{H1s}*N_{H1s}^{T}=(O_4(1,1),\ O_5(1,1),\ O_6(1,1),\ O_8(1,1))(G_{R4}\ G_{R5}\ G_{R6}\ G_{R8})^{T} \\ &=(0,1,1,1)\ (G_{R4}\ G_{R5}\ G_{R6}\ G_{R8})^{T}=G_{R5}+G_{R6}+G_{R8} \end{aligned} \quad (7.3)$$

The H_1 PVS group is the sum of 4 directions, it can be calculated as:

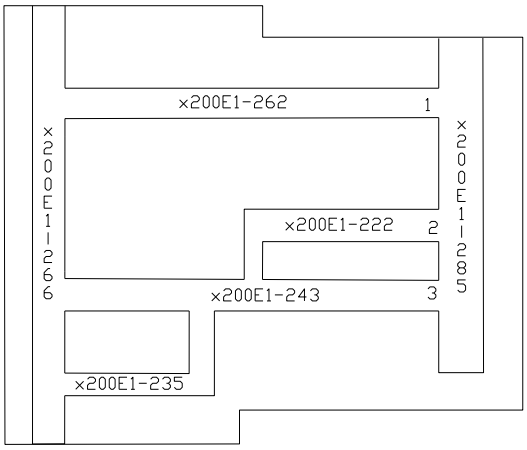
$$G_{H1}=G_{H1n}+G_{H1s}+G_{H1w}+G_{H1e} \quad (7.4)$$

Besides the complete cells included in the PVS group, there are some polygons which is potentially visible to the navigator. Here the decision making is based on the importance

of these polygons to get a reasonable view. For example, some polygons in hallways H_2, H_3, which face the western and eastern sides of the hallway H_1, are included in the PVS group to decrease the effect of sudden switch from one hallway to another.

Removing the repeated nodes from the final group, the whole group was formed and ready to create a dynamic prototype for dynamically loading and unloading in real-time simulation. Figure 7.8 shows the cell segmentation of one floor. The whole model was divided into six PVS regions based on the hallway group. Each region includes its own potential visible cells and potential visible polygons. These regions are linked through entries. For example, entry 1 link x200E1-285 to x200E1-262, and entry 2 link x200E1-285 to x200E1-222, etc. The information is stored in a database.

Figure 1.8 Layout of the E1 2nd floor of EITC building



7.2.4 Data structure of prototypes

The database is used to store a prototype name list for dynamically loading 3D models. EON Planner can dynamically load EON prototypes into an environment by receiving a

filename. By accessing the database directly using EON, we cut out the programming required to send and receive events via the EONX ActiveX Object. The data can be retrieved quickly as there is no need to send and receive events. The interface accesses data by connecting to the database.

Each prototype has a name for loading using a loadnameindex, indicating different entry to the adjacent region. For example, when the current region is x200E2-285, it has three entries leading to x200E1-262, x200E1-222 and x200E1-243. If the navigator goes to the entry 1, the region name will be searched and x200E1-262 will be loaded. This enables many regions to be processed. Table 7.1 shows the relationships of these prototypes.

Table 1.1 Prototype relationship table x200E1load

ID	CurrentArea	LoadName	LoadNameIndex
1	x200E1-285	x200E1-262	1
2	x200E1-285	x200E1-222	2
3	x200E1-285	x200E1-243	3
4	x200E1-262	x200E1-266	1
5	x200E1-262	x200E1-285	2
6	x200E1-266	x200E1-262	3
7	x200E1-266	x200E1-243	1
8	x200E1-266	x200E1-235	2
9	x200E1-235	x200E1-266	2
10	x200E1-235	x200E1-243	1
11	x200E1-243	x200E1-266	3
12	x200E1-243	x200E1-285	1
13	x200E1-243	x200E1-222	2

14	x200E1-222	x200E1-285	1
15	x200E1-222	x200E1-243	2

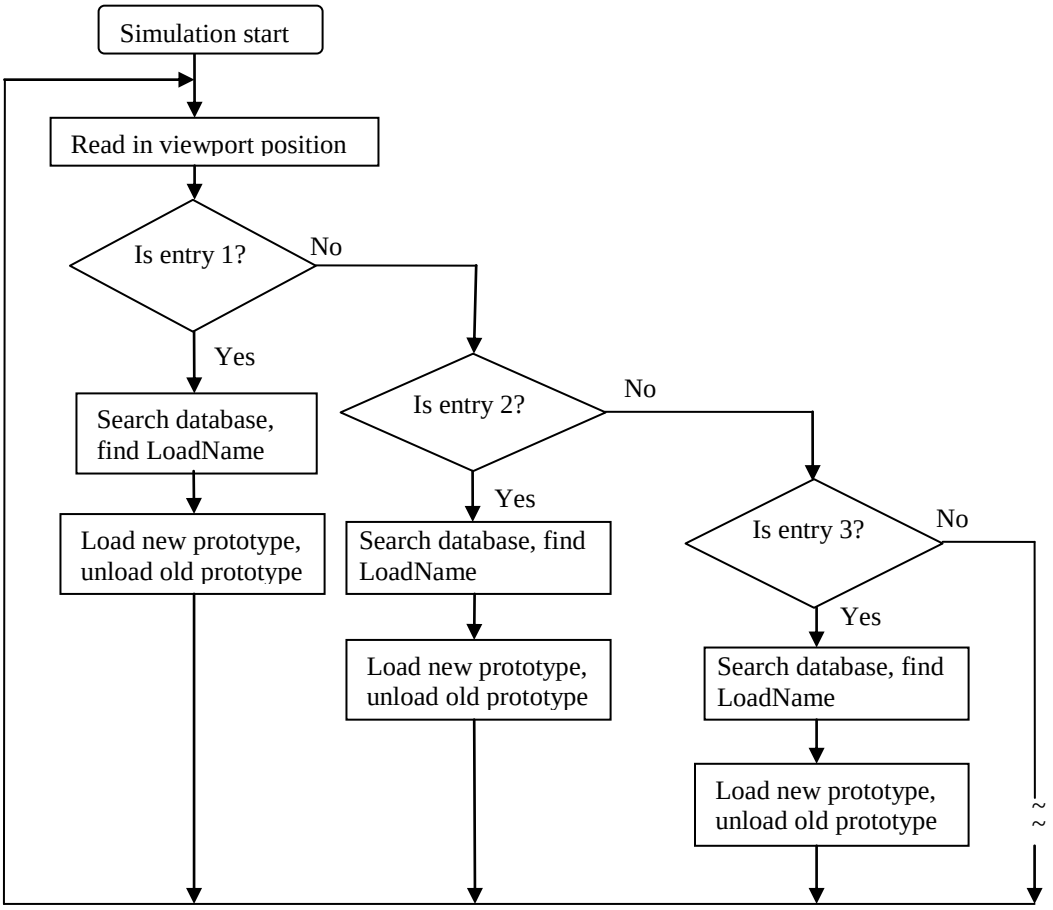
7.2.5 Dynamic loading strategy

EON Dynamic Load is used to load, unload the prototypes dynamically during run-time simulation. All dynamically loaded files are stored in a prototype library. *DynamicPrototype* node can load and view any EON prototype dynamically. This node has a *PrototypeName* field that holds the name of the prototype library file. There is a default prototype, which will be loaded upon the start of the simulation.

During the run-time simulation, the *PrototypeName* field can use the corresponding prototype filename, which means that a new prototype file will be downloaded and inserted to the running EON application. The downloading order is controlled by using script and track current viewport position. When the navigator moves to the entry of another region, the system will search the database and find out which region should be loaded. In the meantime, the previous region is unloaded.

Figure 7.9 shows the flowchart of the dynamic loading. The position of the viewport is sent to a script. In the script, whenever it receives a new position (x, y, z), it will determine whether this position relates to an entry, if yes, it will search the corresponding prototype name (*LoadName* in Table 7.1) in the database according to current prototype and the entry number. The new prototype will then be loaded and old prototype will be unloaded. If the position is not located in any entry, the process will continues.

Figure 1.9 Flowchart of the dynamic loading



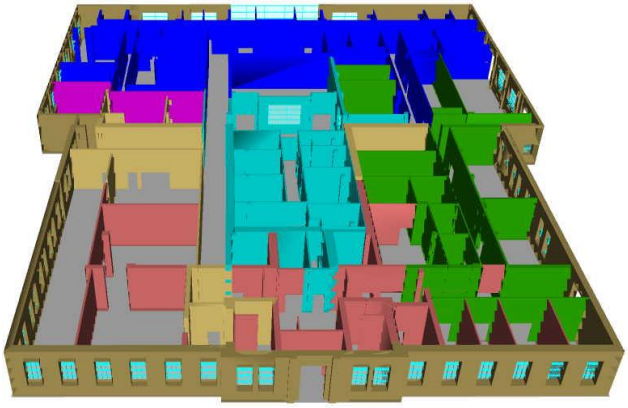
7.2.6 Performance comparison in building navigation

A virtual environment with poor performance is unconvincing, hard to use, and difficult to engage a user into the simulation (Just, 2000). One of the performance measurements in VEs is frame rate. Frame rate is the refreshed frequency of an image in an imaging device, and often expressed in frames per second (fps). If the frame rate is too low, the scene will lag and flash between frames leading to unsatisfied effects. Frame rate is affected by the number of polygons of a geometry model in a VR simulation. In this re-

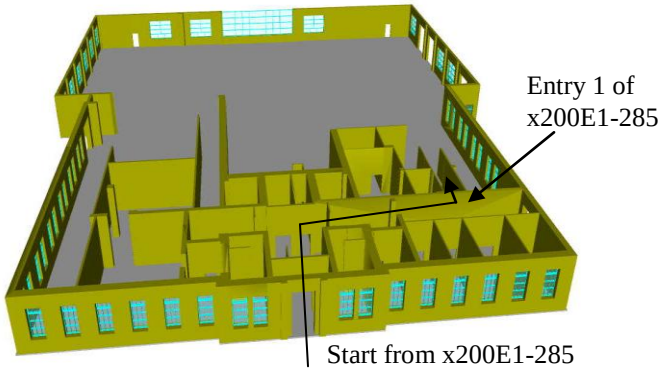
search, a cell segmentation method is used to reduce the number of polygons to be loaded and increase the frame rate in simulation. The number of polygons and frame rate are measured to evaluate the performance improvement using the cell segmentation method.

The proposed method has been tested on the EITC Building at the University of Manitoba using EON studio. The second floor in E1 Building, comprised of 23,852 polygons, was segmented and then grouped into six PVS regions according to the cell properties. Figure 7.10 shows the segmented one floor plan and the dynamic loading. Figure 7.10 (a) illustrates the six segmented regions, in which some polygons are overlapped in two adjacent regions to avoid sudden change of the scene. Figure7.10 (b) shows the external walls and default region x200E1-285 when the simulation starts running. Figure 7.10 (c) illustrates the dynamically loaded region x200E1-262 after the navigator walks to entry 1. The first region is then unloaded. Some separate polygons are included in each region because these walls are visible to the navigator through windows.

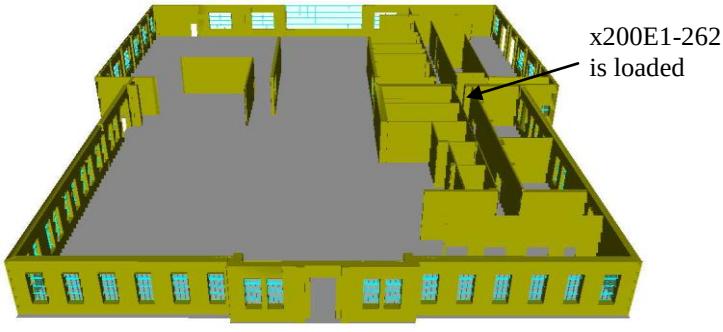
Figure 1.10 (a) The segmented E1 2nd floor plan (b) the front hallway x200E1-285 (c) the dynamically loaded hallway x200E1-262



(a)



(b)



(c)

Table 1.2 Comparison of the number of polygons and frame rate for non-segmented and segmented models

Models of Building		Number of Polygons	Reduction Rate	Frame rate (fps)	Frame rate increased
E1	Non-segmented model	181,408	-	62	-
	Segmented model	60,472	67%	106	71%
E2	Non-segmented model	297,124	-	49	-
	Segmented model	65,976	78%	92	88%

Table 7.2 shows a comparison of non-segmented models and segmented models of Building E1 and E2. The test is conducted in a HP XW4400 Workstation, with Intel (R) Core (TM) 2 CPU 6700@2.66 GHz, and 2.0GB RAM. The segmented model is simplified into a smaller model that only includes external walls, atrium, ceilings, railings, and a current default region. Separated region is dynamically loaded according to the navigator's position. The number of polygons is read from the *PolygonCountAfterReduction* in the *Meshes* node. The frame rate is read from the *StatisticData* under *Simulayion* node. For Building E1, the number of polygons in non-segmented model is 181,408, while the number of polygons in segmented model decreases to 60,472. The reduction rate of E1 is 67%. The frame rate increases by 71%, from 62 fps to 106 fps. For Building E2, the number of polygons in non-segmented model is 297,124, while the number of polygons in segmented model decreases to 65,976. The reduction rate of E2 is 78%. The frame rate increases by 88%, from 49 fps to 92 fps. It shows that the simulation performance is improved after using cell segmentation method.

It seems that the improvement of the navigation speed is not very obvious for only one floor model using the dynamic loading. The frame rate of non-segmentation models is 84Hz, while the frame rate of segmented models with dynamic loading is 86Hz, only 2% improvement in performance. This is not surprising for relatively small models that are tested. Although the region is smaller than the whole model, the real-time monitoring of the viewport position takes time, leading to lower performance than what is expected. However, the frame rate is improved significantly for large models. By loading separate regions, the number of polygons in simulation can be greatly decreased, while the monitoring influence remains the same. The synthesized effect is an improved performance. For the non-segmented models of E2 Building, the frame rate in the simulation is only 17Hz in a computer with low configuration. By using the cell segmentation method, the frame rate of segmented models increases dramatically than non-segmentation models.

7.3 Case study 2: Dynamic loading for a complex product assembly simulation

7.3.1 Overview of dynamic loading

It is computational demanding for the simulation of large product assembly models in VEs, such as a detailed aircraft model, a ship or a mine machine. The loading time may be too long if many small objects are to be loaded simultaneously. Effective rendering and manipulation techniques are required to overcome such computational demanding.

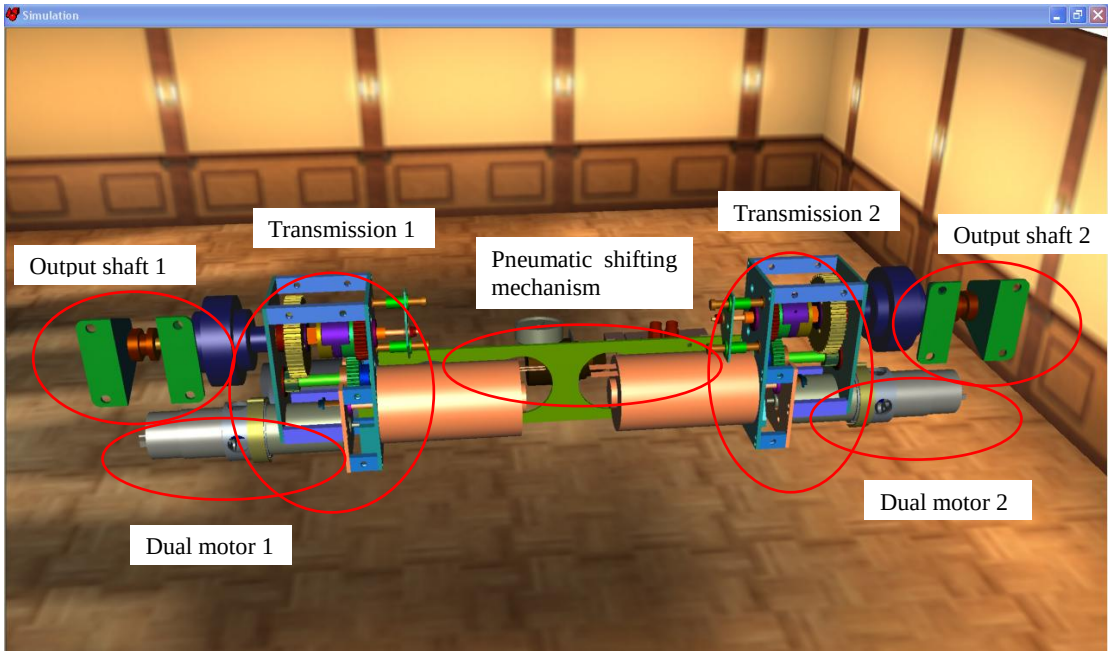
Cell segmentation and dynamic loading method is used in this case study for large product assembly models to improve the VE navigation performance.

Similar to the building navigation, a user may observe details of a small portion of the whole product assembly at a time. Many unseen objects may be unloaded dynamically to reduce the computer memory usage so that the navigation performance is improved. The following section describes similar cell segmentation and dynamic loading strategies applied to the product assembly simulation.

A product assembly model called six motor drive assemblies with pneumatic shifting of 4:1 gearbox (Baker, 2002) is used as an example. It consists of 52 parts. These parts were modeled in Pro/Engineer and exported into EON Studio.

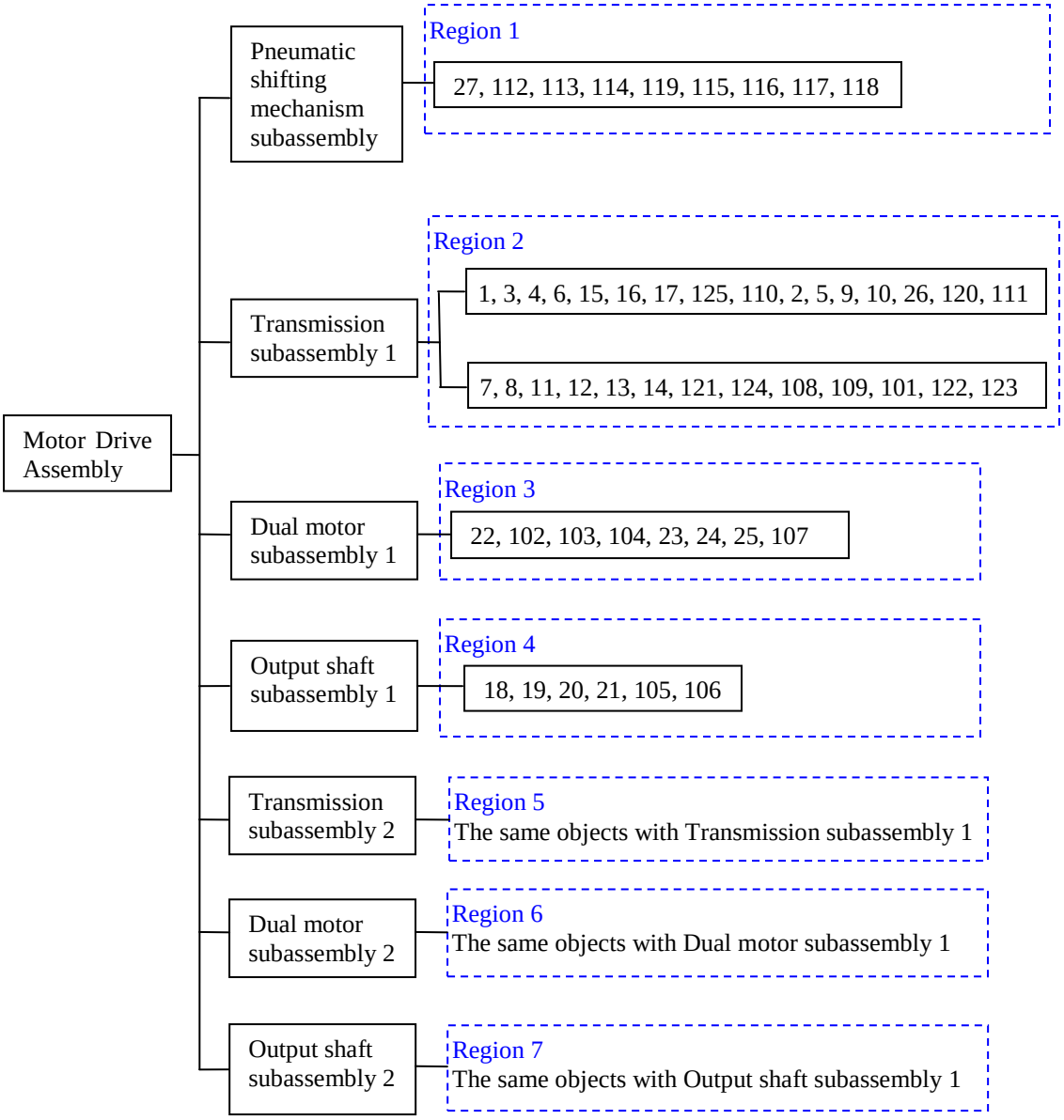
In the pre-processing stage, the product assembly is divided into subassemblies based on its function and structure. The motor drive assembly is divided as pneumatic shifting mechanism, transmission, dual motor, and output shaft subassembly, where the last three subassemblies are located in the left and right sides. There are seven subassemblies in total. A subassembly consists of a group of parts. Figure 7.11 shows the product assembly model and its seven subassemblies.

Figure 1.11 Motor drive assembly in EON Studio



Assume each subassembly is defined as a region, the region division based on subassemblies and their constituted parts are shown in Figure 7.12.

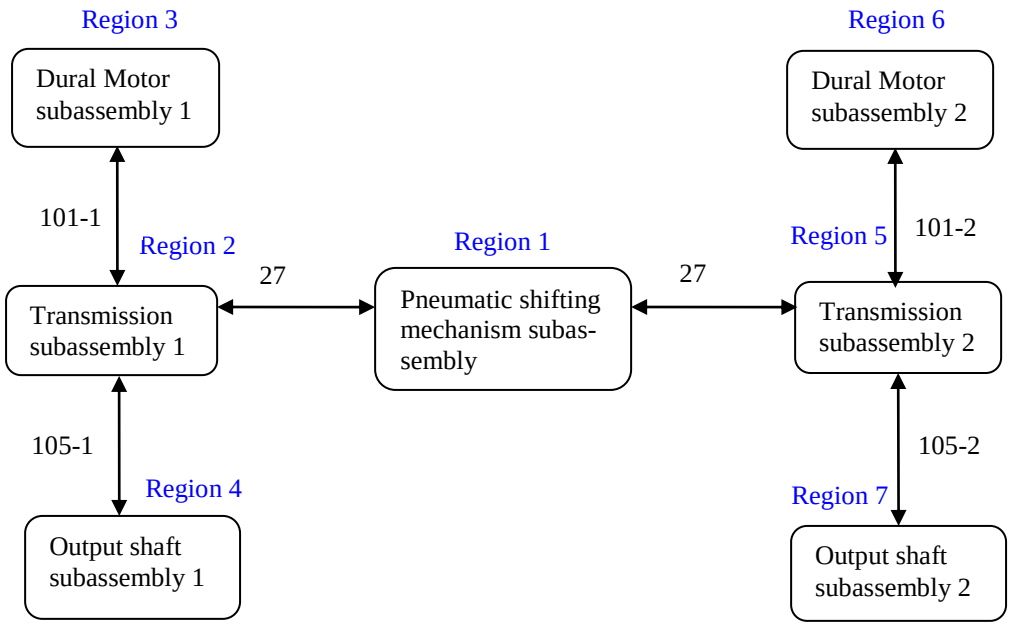
Figure 1.12 Region division and its constituted parts



Two subassemblies may be connected to each other by one or several parts. The connecting parts are essential to show the adjacency of these regions. Figure 7.13 shows the adjacency graph of these regions by connecting parts. For example, region 1 is connected

with region 2 by part 27. The connecting parts are included in the initial product assembly to demonstrate the adjacency between regions.

Figure 1.13 Adjacency graph representation of subassemblies



7.3.2 Cell segmentation

A part is defined as a cell in a region. In order to provide a general product assembly, the initial scene includes a simplified product assembly consisting of several key parts. The key parts are identified based on its profile and function. The parts forming the outer profile of a subassembly and the connection parts are identified as key parts. For example, three key parts are identified in region 1 including parts 115 and 27. Part 115 is assigned as a key part because of its big size. Part 27 is chosen since it's a connecting part. All key

parts in each region form a simplified product assembly model for the initial display. A click sensor is attached to some key parts for responding to a user's action.

7.3.3 Forming regions based on subassemblies

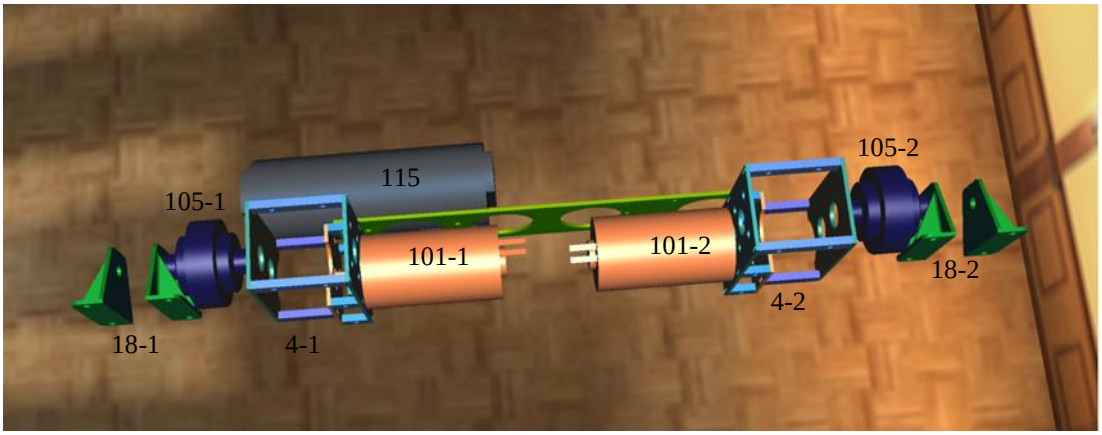
A region is determined based on the parts included in a subassembly. Since key parts have been included in a simplified product assembly model, they are excluded from the region formation. The remaining parts in each region is saved in a prototype file (*.eop) and stored in a prototype library in EON Studio. Table 7.3 shows all key parts identified in each region and the prototypes generated.

Table 1.3 Key parts in the simplified model and prototype forming

Region No.	Key parts	Parts in Prototype	Prototype Name
R1	115, 27	The rest parts of R1	R1.eop
R2	1-1, 2-1, 3-1, 4-1, 5-1, 6-1, 101-1	The rest parts of R2	R2.eop
R3	None	All parts of R3	R3.eop
R4	18-1, 105-1	The rest parts of R4	R4.eop
R5	1-2, 2-2, 3-2, 4-2, 5-2, 6-2, 101-2	The rest parts of R5	R5.eop
R6	None	All parts of R6	R6.eop
R7	18-2, 105-2	The rest parts of R7	R7.eop

These key parts form a simplified product assembly model as shown in Figure 7.14. As we can see, the details of each subassembly are omitted from the simplified product model. But the main structure and connection relationships are clear. A user may proceed to examine the detail structure of a subassembly by clicking the key parts.

Figure 1.14 Simplified product assembly model with key parts



The creation of prototypes follows five steps: (1) Import models into EON Studio, configure the materials and actions. (2) Copy relevant nodes into the same model frames, such as mesh, and materials. (3) Right click the model frame, select “Create Prototype”, a prototype will be created. (4) In the New Component window, click New, save it as prototype file *.eop. (5) Move the generated prototype from local Prototype window to New Component window.

7.3.4 Dynamic loading strategy

During the model simulation, the initial scene displays a simplified product assembly without details of most parts. Only key parts are included in the simplified product assembly. These key parts are clickable. When a key part is clicked by a user, details of the corresponding region will be dynamically loaded. In the meantime, the scene will navigate automatically to a closer viewport for observation.

Table 7.4 shows the relationships of key parts and their dynamic loading prototypes. For instance, if a user clicks the part 115 in the simplified product assembly, the corresponding prototype "R1.eop" will be loaded. The scene will automatically move to that region for better observation. When a user clicks part 115 again, the scene will restore to its initial simplified product assembly model. As soon as another key part is clicked, current prototype "R1.eop" will be dynamically replaced with a new prototype.

Table 1.4 The relationship of key parts and dynamic loading prototypes

Part Name	Prototype Name
115	R1.eop
4-1	R2.eop
101-1	R3.eop
18-1	R4.eop
4-2	R5.eop
101-2	R6.eop
18-2	R7.eop

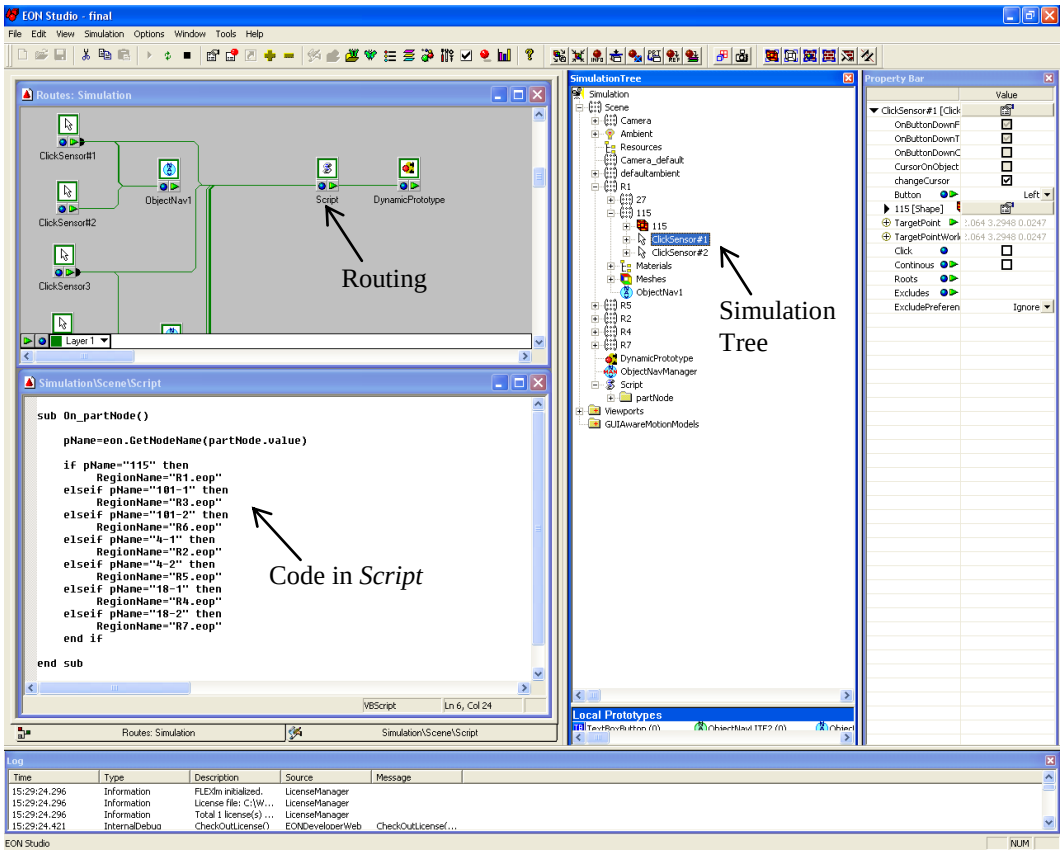
The dynamic loading is implemented using a *Script* node and a *DynamicPrototype* node. The *ClickSensor* sends the part name to *script*. In *Script*, the required loading region is found based on the clicked part. This region name is then sent to the *PrototypeName* field of *DynamicPrototype* node, and the prototype is loaded automatically. The routing is as follow:

ClickSensor.Target->Script.ObjectName

Script.RegionName-> DynamicLoading.PrototypeName

When a subassembly is loaded, the scene will automatically navigate to the newly loaded subassembly. An *ObjectNav* prototype is assigned to each subassembly. *ObjectNav* is a prototype that allows a user to flip (rotate), zoom, and pan a 3D object using mouse or mouse and key combinations. It automatically resets the camera when *ObjectNav* becomes active by presetting the camera's coordinates. *ObjectNavManager* prototype controls the shifting between *ObjectNavs*. It turns off the previous *ObjectNav* when a new *ObjectNav* has become active. Figure 7.15 shows the system implementation in EON Studio. The left side is the routing between different nodes and the coding of *Script* node. The middle is the simulation tree of all nodes and models.

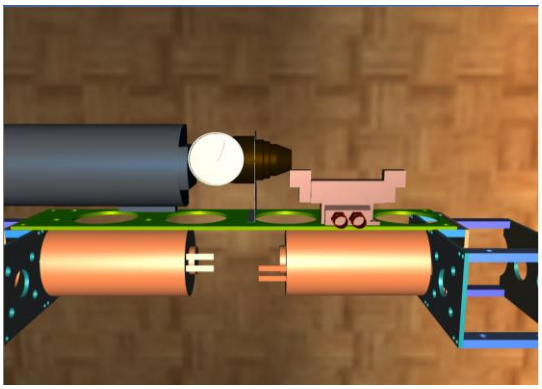
Figure 1.15 The implementation of dynamic loading in Eon Studio



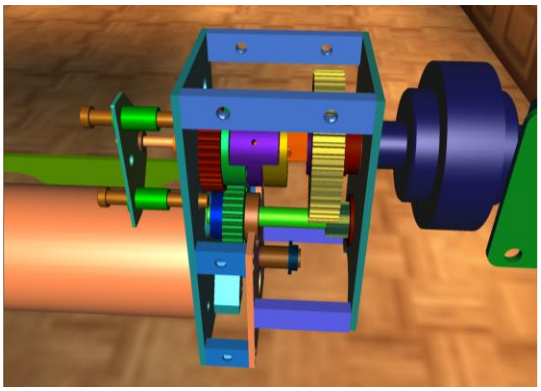
7.3.5 Performance comparison in product assembly simulation

The product assembly is implemented according to the above methods. The result shows performance improvements based on the number of polygons and frame rate in the real-time simulation. Figure 7.16 shows the dynamically loaded regions based on a user's selection. In Figure 7.16 (a), the parts in region 1 are dynamically loaded when part 115 is clicked. In Figure 7.16 (b), the initial transmission model is a frame without the inside

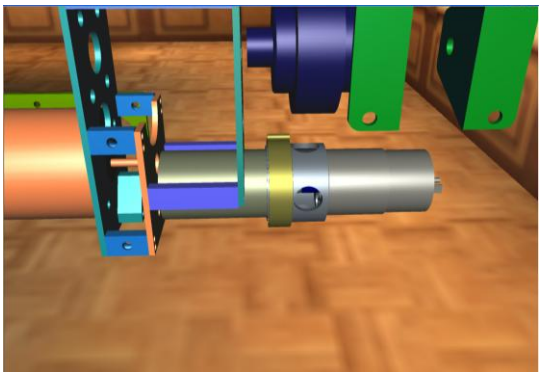
Figure 1.16 Four dynamic loaded regions based on a user's selection (a) the transmission (b) the pneumatic shifting mechanism (c) the dual motor (d) the output shaft



(a)



(b)



(c)

(d)

parts. When part 4-2 is clicked, the parts in region 5 are loaded including gears, axes, and dog mate. Figure 7.16 (c) shows the loaded dual motors. Figure 7.16 (d) shows the loaded parts in output shaft 2.

Table 7.5 shows a comparison of the number of polygons and frame rate before and after cell segmentation. The test is performed in a Dell Workstation PWS530, with Intel(R) Xeo CPU 2.0 GHz, and 523KB RAM. There are 65,484 polygons in the original assembly. The number of polygons is reduced to 32,816 in the simplified assembly, which accounts to 50% reduction of polygons. The frame rate increases from 50 fps to 63fps, a 26% performance improvement. When a region is dynamically loaded, the polygon's reduction rate ranges from 23% to 47%, and the frame rate increases up to 20%, comparing to the original assembly. It demonstrates that cell segmentation and dynamic loading method decreases the number of polygons in the computer's memory, thus the navigation performance is improved.

Table 1.5 Comparison of the number of polygons and frame rate

Models	Number of polygons	Reduction rate	Frame rate (fps)	Frame rate increased
Original assembly	65484	0	50	-
Simplified assembly	32816	50%	63	26%
Simplified assembly + R1	36976	44%	59	18%
Simplified assembly + R2	50610	23%	55	10%
Simplified assembly + R3	42580	35%	57	14%
Simplified assembly + R4	34616	47%	60	20%

7.4 Chapter summary

This chapter presents the construction of large-scale VEs and the development of dynamic loading methods to improve the VE simulation performance. Two case studies are conducted including a building navigation and a product assembly simulation. EON Studio is used as a VR modeling tool. In the building navigation, a region forming method based on the cell segmentation and visibility calculation is developed. Cell properties are used to form regions according to the potentially visibility during simulation. Dynamic loading strategy is applied by real-time monitoring of the viewport position. For the product simulation, regions are formed based on subassembly divisions. Dynamic loading is realised by user's selection on some predefined key parts. The developed methods are very useful in the construction of large-scale VEs. It has the advantages of reduced computer memory usage, improved navigation performance, and support local design modification.

Chapter 8

Conclusions and future work

This chapter summarizes the conducted research in this thesis for dynamic fixture planning. The research contributions are identified, and the future work is recommended.

8.1 Research contributions

This research developed automatic approaches to dynamic fixture planning in virtual manufacturing environments. It proposed effective algorithms for fixture assembly planning, the feasibility analysis of assembly tools, and motion planning for fixture loading and unloading. The construction of Web-based virtual environments and model management techniques in large-scale VEs are discussed for performance improvements. The research integrates fixture environmental factors into the automatic fixture planning to meet dynamic manufacturing operations. The proposed VEs support high fidelity visualization for fixture planning, and other virtual design and manufacturing tasks. The contributions of this research are summarized as follows.

8.1.1 Feasibility analysis of assembly tools

A new approach to test the tool feasibility in fixture assembly planning is developed. Based on a newly defined global accessibility cone of a truncated half-line ($GAC^{d}_{(R)}$), the proposed system can automatically determine the feasibility of an assembly tool in the application. Assembly tools are modeled as five articulated parts with seventeen parameters. The tool application is simulated in a 3D virtual environment to demonstrate the tool feasibility analysis for the fixture assembly. The same tooling consideration can be applied in the fixture disassembly and workpiece loading/unloading. The integration of assembly tools' feasibility analysis with fixture assembly planning is essential for generating a complete fixture assembly plan. It can also be used to efficiently support computer-aided assembly planning (CAAP), design for manufacturing (DFM), design for assembly (DFA), design for disassembly (DFD), and computer-aided tool selection (CATS).

8.1.2 Fixture assembly planning

This research developed a fastener-based approach to fixture assembly planning. The topological geometry constraints, the feasibility of an assembly tool, and fixture functional constraints are analyzed to generate a feasible fixture assembly plan. The developed system can generate fixture assembly/disassembly sequence plans and a feasible assembly tool list. It can reduce the lead time for fixture assembly/disassembly planning and the assembly tool selection. It can also evaluate the feasibility of a fixture assembly or disassembly in the early design stage.

8.1.3 Motion planning for fixture loading and unloading

A new accessibility-based A Star (A*) algorithm for motion planning in fixture loading and unloading is developed. A spherical accessibility matrix is used to identify accessible moving directions of a workpiece based on geometrical reasoning. A* algorithm is integrated with the spherical accessibility matrix to optimize the motion path. The implementation of the proposed algorithm shows the feasibility of the collision-free motion plan. The proposed algorithm is significant in automatic planning of workpiece's loading path. It enables a fast fixture evaluation in fixture loading and unloading, and the dynamic simulation of the generated motion plan.

8.1.4 A Web-based VE for fixture planning

A Web-based VE for fixture planning is generated for the design collaboration and visualization. The Web-based fixture planning system uses a three-tier client/server architecture including a Web server, a DB server and clients. HTML, Java applet and VRML browser are adopted for the user interface on the client side to simulate 3D models of a fixture and generated fixture plan. A Web server and a socket-based communication are used to accept clients' request and to communicate with the DB server. MySQL DBMS is adopted as the database server which stores the data of fixtures and assembly tools. The generated VE supports design collaboration and the visualization of the generated fixture plan.

8.1.5 The construction and performance improvement of large-scale VEs

Cell segmentation techniques are investigated in the construction of large-scale VEs to improve the simulation performance. An improved method of forming several regions based on the potentially visibility is proposed. Unlike the previous methods, a matrix calculation is used to group the PVS, which forms a conservative set of regions in a preprocessing stage. These PVS groups are converted into prototypes and stored in the local disk or a Web server. During a real-time simulation, these prototypes are loaded and unloaded dynamically according to the viewpoint for easy loading of large-scale VR models and fast rendering. Two case studies are developed to test the proposed method including a building model with massive polygons and a product assembly. The result shows that the VE performance is improved comparing to the non-segmented model.

8.2 Future work

Several potential research directions may be continuously explored in the future. More efficient and robust algorithms for fixture planning can be developed and incorporated into this research. The recommendations for future work are summarized as follows:

1. The developed algorithms for fixture planning concentrate on fixture assembly planning, assembly tool analysis, and fixture loading and unloading, rather than fixture design and setup planning. Thus, algorithms for fixture design and setup planning based on geometrical, accessibility, force, and deformation constraints are worthwhile to be studied. The new algorithms can be integrated with this research to provide a complete fixture planning system.

2. For assembly tools' analysis, research could be extended to the motion planning of assembly tools when approaching the fastener, as well as the feasibility analysis of human hands or robot arms grasping the tool. In addition, the tool feasibility analysis for special tools under extreme circumstances, such as automotive maintenance, may be researched. Moreover, since fasteners are often tightened with a certain amount of torque. It may be required to specify the use of tools to overcome the residual torque, such as the point of applications, the minimal tool length, and other relevant parameters. Such force-related parameters may be included in the assembly tool modeling.

3. Some existing research in fixture assembly planning has incorporated intelligent means to aid the automatic robotic assembly. The environmental factors can be better captured and reflected using sensors, cameras, and etc. More realistic and robust algorithms in this direction can be further investigated.

4. Current fixture assembly planning approach only considers the one-step translational motion of the fixture elements. The research can be extended to investigate multi-step translational and rotational movements for the robotic fixture assembly and disassembly planning. Industrial applications should be further tested to verify the feasibility and efficiency of the developed approaches.

5. Current motion planning algorithm is based on a random sampling method for the node expansion. The effect of sampling methods over the efficiency of motion planning is worth to be looked at in order to improve the sampling methods for

the path generation. Current algorithm is limited to accessible directions along a straight line. Some modification to the algorithm is possible to handle non-straight line paths.

6. In the analysis of fixture loading and unloading, the attention should also be paid to the ergonomics analysis and safety issues when human-like or robot operators are integrated into the motion planning. The evaluation criteria can include more heuristic factors, such as the convenience of an operator, to get an easy and convenient motion plan in a fixture loading operation. Immersive assembly and disassembly planning in VR using Head-mounted devices, data gloves, and cyber glasses can be used to verify the generated assembly plan without actually making the objects or parts.

7. The Web-based system can be updated to a new XML-X3D structure so that the developed algorithms are adapted to new technologies.

8. A problem in the dynamic loading for large-scale VEs is the sudden switch between different models when loading a new model. One possible solution is refining the entry position values and overlapped more polygons in the entry area. Another strategy is maintaining two adjacent regions simultaneously, and loading the new model when the navigator approaches the entry. Finally, LOD and other model management techniques can be further developed to improve the method.

Bibliography

Airey, J.-M.. 1990. Increasing update rates in the building walkthrough system with automatic model-space subdivision and potentially visible set calculations. *PhD thesis*. University of North Carolina.

Aliaga, D.-G., A.-A. Lastra, 1997. Architectural walkthroughs using portal textures. *IEEE visualization'97:* 355-362.

Asada, H., A. By. 1985. Kinematic analysis of workpart fixturing for flexible assembly with automatically reconfigurable fixtures. *IEEE Journal of Robotics and Automation* 1: 86–94.

Bai, Y., Y.Rong. 1995. Establishment of modular fixture element assembly relationship for automated fixture design. *ASME Manufacturing Engineering Division, MED, Manufacturing Science and Engineering* 2-1: 805-816, November.

Baker, A.. 2002. 6 Motor drive assembly w/Pnumatic shifting of 4:1 gearbox. developed by the TechnoKats Robotics Team 45, Kokomo, Indiana.

Baker, M. J. 2008. *Math-rotation matrices'*, http://www.euclideanspace.com/maths/algebra/matrix/orthogonal/rotation/index.htm

Bi, Z. M., W. J. Zhang. 2001. Flexible fixture design and automation: review, issues and future directions. *International Journal of Product Research* 39(13): 2867-2894.

Bittner, J., V. Havran, P. Slavik. 1998. Hierarchical visibility culling with occlusion trees. *Proceedings of Computer Graphics International'98*: 207-219.

Boothroyd, G., P. Dewhurst, W. Knight. 1994. Product design for manufacture and assembly. Marcel Dekker Inc., New York.

Boyle, I.. 2006. CAFixD: A case-based reasoning method for fixture design. *Ph.D. thesis*, Worcester Polytechnic Institute.

Brost, R. C., K. Y. Goldberg. 1994. A complete algorithm for synthesizing modular fixtures for polygonal parts. *Proceedings of 1994 IEEE on Robotics and Automation* 1: 535-542.

Canny, J.. 1998. The complexity of robot motion planning. *ACM Doctoral Dissertation Award*, MIT Press.

Carey, R., G. Bell. 1997. The Annotated VRML 2.0 Reference Manual (OpenGL). *Addison-Wesley Professional*.
http://accad.osu.edu/~pgerstma/class/vnv/resources/info/AnnotatedVrmlRef/ch3-352.htm.

Cecil, J.. 2001. Computer-aided fixture design - A review and future trends. *International Journal of Advanced Manufacturing Technology* 18(11): 790-793.

Cecil, J.. 2004. TAMIL: an integrated fixture design system for prismatic parts. *International Journal of Computer Integrated Manufacturing* 17(5): 421-34.

CGTech Ltd.. http://www.cgtech.com/usa/index.php, cited on December 24, 2007.

Chen, G.-F., W.-J. Liu. 2002. Variant fixture design with CBR. *Proceedings of 2002 International Conference on Machine Learning and Cybernetics (Cat.No.02EX583)* 3(3): 1465-1469.

Chung, C., Q. Peng, Q., 2006-1. Evolutionary sequence planning for selective-disassembly in de-manufacturing. *International Journal of Computer Integrated Manufacturing* 19(3): 278-286.

Chung, C., Q. Peng. Q. 2006-2. A hybrid approach to selective-disassembly sequence planning for de-manufacturing and its implementation on the Internet. *International Journal of Advanced Manufacturing Technology* 30: 521-529.

Chung, C., Q. Peng. 2006-3. A novel approach to the geometric feasibility analysis for fast assembly tool reasoning. *International Journal of Advanced Manufacturing Technology*. 31: 125-134.

Cicek, A., M. Gulesin. 2007. A part recognition based computer aided assembly system. *Journal of Computers in Industry* 58(8-9): 733-746.

Cohen-Or, D., G. Fibich, D. Halperin, E. Zadicario. 1998. Conservative visibility and strong occlusion for viewspace partitioning of densely occluded scenes. *Computer Graphics Forum* 17/3: 243-254.

Coorg, S., S. Teller. 1996. Temporally coherent conservative visibility. *Proceedings of 12th Annunal ACM Symposium Computer Geometric*: 78-87.

Dai, J. R., , A. Y. C. Nee, J. Y. H Fuh,., A.Senthil Kumar. 1997. An approach to automating modular fixture design and assembly. *Proceedings of the Institution of Mechanical Engineers, Journal of Engineering Manufacturing* Part B, 211(7): 509-21.

Dechter, R., J. Pearl. 1985. Generalized best-first search strategies and the optimality of A*. *Journal of the Association for Computing Machinery* 32(3): 505-536.

Diehl, S.. 2001. Distributed Virtual Worlds-Foundations and Implementation Techniques Using VRML, Java, and CORBA, Springer-Verlag, Berlin.

Dong, J., G. Arndt. 2003. A review of current research on disassembly sequence generation and computer aided design for disassembly. *Proceedings of the Institute of Mechanical Engineers, Part B. Journal of Engineering Manufacture* 217(3): 299-312.

Fan, L., A. S. Kumar. 2005. XML-based Representation in a CBR System for Fixture. Design. *Computer-Aided Design and Applications* 2(1-4): 339-348.

Fuh, J. Y. H., A. Y. C. Nee, A. S. Kumar, J. C. S. Teo. 1995. IFDA: an interactive fixture design and assembly environment. *International Journal of Computer Application Technology*.8(1/2): 30–40.

Gelautz, M., M. Brandejski, F. Kilzer, F. Amelung. 2004. Web-based visualization and animation of geospatial data using X3D. *2004 IEEE International Proceedings on* Geoscience and Remote Sensing Symposium(IGARSS '04).7: 4773 - 4775.

Greene, N., M. Kass, G. Miller. 1993. Optimized occlusion culling. *Computer & Graphics* 23/5: 43-54.

Gupta, S., C. Paredis, P.F. Brown. 1998. Micro Planning for Mechanical Assembly Operations. *Proceedings of IEEE International Conference on Robotics and Automation* 1: 239 - 246.

Hamedi, M.. 2005. Intelligent fixture design through a hybrid system of artificial neural network and genetic algorithm. *Artificial Intelligence Review* 23: 295–311.

Hargrove, S. K., A. Kusiak. 1994. Computer-aided fixture design: a review. *International Journal of Production Research* 32(4): 733-753.

Hazen, F., P. Wright. 1990. Workholding automation, innovations in analysis, design and planning. *Manufacturing Review* 3 (4): 224-237.

Hou, J. L., A. J. C. Trappey. 1997. A Methodology for applying V-blocks and clamps to non-prismatic workpart fixtures. *International Journal of Computer Applications in Technology* 10(3-4): 152-169.

Hou, J.-L., A. J. C. Trappey. 2001. Computer-aided fixture design system for comprehensive modular fixtures. *International Journal of Production Research* 39(16): 3703-3725.

Hren, G., A. Jezernik, B. Zalik. 2006. Web-based framework for bidirectional link of VRML and CAD assemblies. *International Journal of Product Lifecycle Management* 1(3): 303-320.

Hu, W., Y. Rong. 2000. A Fast Interference Checking Algorithm for Automated Fixture Design Verification. *International Journal of Product Research* 16: 571-581.

Huang, Y. M., C. T. Huang. 2002. Disassembly matrix for disassembly processes of products. *International Journal of Production Research* 40(2): 255-73.

Hunter, R., A. Vizan, J. Perez, J. Rios. 2005. Knowledge model as an integrated way to reuse the knowledge for fixture design process. *Journal of Materials Processing Technology* 164-165: 1510-1518.

Ilushin, O., G. Elber, D. Halperin, R. Wein, M. S. Kim. 2005. Precise Global Collision Detection in Multi-axis NC-machining. *Computer-Aided Design* 37(9): 909-920.

Jackman, J., D. K. Park. 1998. Probe Orientation for Coordinate Measuring Machine Systems Using Design Models. *Robotics and Computer-Integrated Manufacturing* 14: 229-236.

Jang, H. Y., H. Moradi, S. Lee, J. H. Han. 2005. A visibility-based accessibility analysis of the grasp points for real-time manipulation. *2005 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2005:* 3111-3116.

Jayaram, U., Y. J. Kim, S. Jayaram, V. K. Jandhyala, T. Mitsui. 2004. Reorganizing CAD assembly models (as-designed) for manufacturing Simulations and planning (as-built). *Transactions of the ASME, Journal of Computing and Information Science in Engineering* 4(2): 98-108.

Joneja, A., T.-C. Chang. 1999. Setup and fixture planning in automated process planning systems. *IIE transactions* 31(7: 653-665.

Christopher Derek Just, Performance analysis of a virtual reality development environment: Measuring and tooling performance of VR Juggler. *Master thesis*, Iowa State University, 2000.

Kakish, J., P. Zhang, I. Zeid. 2000. Towards the design and development of a knowledge-based universal modular jigs and fixtures system. *Journal of Intelligent Manufacturing* 11: 381-401.

Kallmann, M., A. Aubel, T. Abaci, D. Thalmann. 2003. Planning collision-free reaching motions for interactive object manipulation and grasping. *EUROGRAPHICS 2003* 22(3): 313-322.

Kang, Y., Y. Rong, J. C. Yang. 2003. Computer-aided fixture design verification: Part 1. the framework and modeling. *International Journal of Advanced Manufacturing Technology* 21(10-11: 836-841.

Kang, Y. G., Z. Wang, R. Li, C. Jiang. 2007. A fixture design system for networked manufacturing. *International Journal of Computer Integrated Manufacturing* 20(2): 143-159.

Kaya, N.. 2006. Machining fixture locating and clamping position optimization using genetic algorithms. *Computers in Industry* 57: 112-120.

Kumar, A. S., J. Y. H. Fuh, T. S. Kow. 2000. An automated design and assembly of interference-free modular fixture Setup. *Computer-Aided Design* 32: 583–596.

Lavalle, S. M.. 2006. Planning algorithms. Cambridge University Press.

Lazzerini, B., F. Marcelloni. 2000. A Genetic Algorithm for Generating Optimal Assembly Plans. *Artificial Intelligence in Engineering* 14: 319-329.

Leyvand, T., O. Sorkine, D. Cohen-Or. 2003. Ray space factorization for from region visibility. *ACM SIGGRAPH 2003*: 595-604.

Li, M., S. Gao, C. L. Wang. 2007-1. Real-time collaborative design with heterogeneous CAD systems based on neutral modeling commands. *ASME transactions, Journal of Computing and Information Science in Engineering* 7: 113-125.

Li, W. D., Y. L. Cai, W. F. Lu. 2007-2. A 3D simplification algorithm for distributed visualization. *Journal of Computers in Industry* 58(3): 211-226.

Li, J., W. Ma, Y. Rong. 1999. Fixturing Surface Accessibility Analysis for Automated Fixture Design. *International Journal of Product Research* 37(13): 2997-3016.

Li, W., P. Li, Y. Rong. 2002. Case-based agile fixture design. *Journal of Materials Processing Technology* 128: 7–18.

Li Qiang. 2008. Virtual reality for fixture design and assembly. *PhD thesis*, The University of Nottingham, England.

Lim, C. P., C. H. Menq. 1994. CMM Feature Accessibility and Path Generation. *Robotics and Computer Integrated Manufacturing* 32 (3): 597-618.

Limaiem, A., H. A. ElMaraghy. 1997. A General Method for Analyzing the Accessibility of Features Using Concentric Spherical Shells. *International Journal of Advanced Manufacturing Technology* 13: 101-108.

Liu, C., Y. Zhang, L. Sun. 2008. Web based 3D assembly sequence planning prototype integrated with CAD model. *12th International Conference on Computer Supported Cooperative Work in Design, (CSCWD 2008)*: 823- 828. 16–18 April, Xi'an, China.

Liu, Y.-H., M. L. Lam, D. Ding. 2004. A complete and efficient algorithm for searching 3-D form-closure grasps in the discrete domain. *IEEE Transactions on Robotics* 20(5): 805-816.

Liu, Y. H.. 2004. Optimal Fixture Layout Design for 3-D Workpieces. *Proceedings of the 2004 IEEE International Conference on Robotics 8 Automation* 5: 5274-5279.

Liu, X., X. Zhang, W. Liu. 2007. Integration of CAPP and CAFD based on agent technology. *Proceedings of International Conference on Mechatronics(ICM2007).* Kumamoto Japan.

Lozano-Perez, T. 1997, A simple motion-planning algorithm for general robot manipulators. *IEEE Journal of Robotics and Automation* RA-3(3), June.

Lu, L., S. Akella. 2000. Folding cartons with fixtures: a motion planning approach. *IEEE Transactions on Robotics and Automation* 16(4): 346-56, August.

Luebke, D., C. Georges. 1995. Portals and mirrors: Simple, fast evaluation of potentially visible sets. *Proceedings of the Symposium on Interactive 3D Graphics*: 105-106.

Ma, W., Z. Lei, Y. Rong. 1998. FIX-DES: A Computer-Aided Modular Fixture Configuration Design System. *International Journal of Advanced Manufacturing Technology* 14: 21-32,

Mascle, C., B.-A. Balasoiu. 2003. Algorithmic selection of a disassembly sequence of a component by a wave propagation method. *Robotics and Computer Integrated Manufacturing* 19: 439-448.

Mazer, E., J. M. Ahuactzin, P. Bessiere. 1998. The Ariadne's clew algorithm. *Journal of Artificial Intelligence Research* 9: 295-316.

Mervyn, F., A. S. Kumar, S. H. Bok, A. Y. C. Nee. 2003. Development of an internet-enabled interactive fixture design system. *CAD Computer Aided Design* 35(1): 945-957.

Mervyn, F., A. S. Kumar, A.Y.C. Nee. 2006. Fixture design information support for integrated design and manufacturing. *International Journal of Production Research* 44(11): 2205-2219.

Moore, K. E., A. Güngör, S. M. Gupta. 2001. Petri net approach to disassembly process planning for products with complex AND/OR precedence relationships. *European Journal of Operational Research* 135: 428-449.

Pan, C., S. F. Smith, G. C. Smith. 2005. Determining interference between parts in CAD STEP files for automatic assembly planning. *Transactions of the ASME. Journal of Computing and Information Science in Engineering* 5(1): 56-62.

Pan, C., S. Smith. 2003. Extracting geometrical data from CAD STEP files. *Proceedings of DETC:* 503-508. Chicago, USA.

Pehlivan, S., J. D. Summers. 2008. A review of computer-aided fixture design with respect to information support requirements. *International Journal of Production Research* 46(4): 929-947.

Peng, G., J. Gao, X. He. 2009. Towards the development of a desktop virtual reality-based system for modular fixture configuration design. *Assembly Automation* 29(1). DOI 10.1108/01445150910929820.

Peng, G., W. Liu. 2006. A novel modular fixture design and assembly system based on VR. 2006 *IEEE/RSJ International Conference on Intelligent Robots and Systems*: 2650-5.

Peng, Q., C. Chung. 2007. Analysis of part accessibility in product disassembly. *Journal of Computer-aided Design and Applications* 4(5): 695-704.

Pimentel. K., K. Teixeira. 1993. Virtual Reality: Through the New Looking Glass. Windcrest McGraw-Hill, New York.

Rajan, V. N., K. Sivasubramanian, J. E. Fernandez. 1999. Accessibility and ergonomic analysis of assembly product and jig designs. *International Journal of Industrial Ergonomics* 23: 473-487.

Roden, T., I. Parberry. 2005. Portholes and Planes: Faster Dynamic Evaluation of Potentially Visible Sets. *ACM Computers in Entertainment (CIE)* 3/2:1-9.

Roy, U., J. Liao. 1998. Application of a blackboard framework to a cooperative fixture design system. *Computers in Industry* 37: 67–81.

Roy, U., S. S. Kodkani. 1999. Product modeling within the framework of the World Wide Web. *IIE Transactions* 31: 667–77.

Saona-Vazquez, C., I. Navazo, P. Brunet. 1999. The visibility octree: A data structure for 3D navigation. *Computer & Graphics* 23/5: 635-643.

Schimmels, J. M., M. A. Peshkin. 1992. Admittance matrix design for force-guided assembly. *IEEE Transactions on Robotics and Automation* 8(2): 213–227. April.

Schwartz, J. T., M. Sharir. 1983. On the piano movers problem ii, general techniques for computing topological properties of real algebraic manifolds. *Advances of Applied Maths* 4: 298.351.

Spitz, S. N., A. J. Spyridi, A. A. G., Requicha. 1999. Accessibility analysis for planning of dimensional inspection with coordinate measuring machines. *IEEE Transactions on Robotics and Automation* 15 (4): 714-727.

Spitz, S. N., A. A. G. Requicha. 2000. Accessibility Analysis Using Computer Graphics Hardware. *IEEE Transactions on Visualization and Computer Graphics* 6 (3): 208-219.

Spyridi, A. J., A. A. G. Requicha. 1990. Accessibility Analysis for the Automatic Inspection of Mechanical Parts by Coordinate Measuring Machines. *Proceedings of IEEE International Conference on Robotics and Automation*: 1284-1289.

Srinivasan, H., R. Gadh. 1998. A geometric algorithm for single selective disassembly using the wave propagation abstraction. *Computer-Aided Design* 30(8): 603-613.

STEP Tools, Inc. 2008. ST-Viewer 5.1 overview.
http://www.steptools.com/products/stviewer/overview.html.

Sundaram, S., I. Remmler, N. M. Amato. 2001. Disassembly sequencing using a motion planning approach. *Proceedings of the 2001 IEEE International Conference on Robotics & Automation*. May 21-26, 2001, Seoul, Korea.

Teller, S.-J., C.-H. Seaquin. 1991. Visibility preprocessing for interactive walkthroughs. *Computer graphics* 25/4: 61-69.

Tseng, Y.-J.. 1998. Feature-based fixturing analysis for machining parts represented with multiple sets of features. *International Journal of Product Research* 36(10): 2743-2770.

Trappey, A. J., C. R. Liu. 1990. A literature survey of fixture design automation. *International Journal of Advanced Manufacturing Technology* 5: 240-55.

Varadhan, G., D. Manocha. 2005. Star-shaped roadmaps - a deterministic sampling approach for complete motion planning. *Proceedings of Robotics: Science and Systems*. Cambridge, USA, June.

Wagner, R., G. Castanotto, K. Goldberg. 1997. FixtureNet: Interactive computer-aided design via the world wide web. *International Journal of Human-Computer Studies* 46: 773-788.

Wang, M. Y.. 2000. An optimum design for 3-D fixture synthesis in a point set domain. *IEEE Transactions on Robotics and Automation* 16(6): 839-846.

Wang, H., D. Xiang, G. Duan, L. Zhang. 2007. Assembly planning based on semantic modeling approach. *Journal of Computers in Industry* 58: 227–239.

Wang, H., Q. Niu, D. Xiang, G. Duan, G., 2006. Ant colony optimization for disassembly sequence planning. *Proceedings of 2006 ASME International Design Engineering Technical Conference & Computers and Information in Engineering Conference*. September 10-13, Philadelphia, Pennsylvania, USA.

Wang, L., B. Wong, W. Shen, S. Lang. 2002. Java 3D enabled cyber workspace. *Communications of the ACM* 45(11): 45-49.

Wang, Y., X. Chen, Q. Liu, N. Gindy. 2006. Optimisation of machining fixture layout under multi-constraints. *International Journal of Machine Tools & Manufacture* 46: 1291–1300.

Web3D Consortium. 2007. Available at: http://www.web3d.org/x3d/specifications. Accessed in July 2008.

Wilson, R. H.. 1998. Geometric reasoning about assembly tools. *Artificial Intelligence* 98 (1-2): 237-279.

Wonka, P., M. Wimmer, D. Schmalstieg. 2000. Visibility preprocessing with occluder fusion for urban walkthrough. *Rendering Techniques 2000: 11th Eurographics Workshop on Rendering*: 71-82.

Wu, Y. Y.. 2001. The geometric principle of automatic modular fixture planning. *Proceedings of the 2001 ASME Design Engineering Technical Conference and Computers and Information in Engineering conference* 1: 779-787.

Yi, C., G. A. Nekey. 1996. Assembly planning for modular fixtures. Proceedings of the 1996 IEEE/RSJ International Conference on Intelligent Robots and Systems. *IROS 96. Robotic Intelligence Interacting with Dynamic Worlds (Cat. No.96CH35908)* 2(2): 704-711.

Yu, J. C., Y. M. Li. 2006. Structure representation for concurrent analysis of product assembly and disassembly. *Expert Systems with Applications* 31(4): 705-714.

Yu, J.-C., Y.-M. Li. 2005. The structure representation for the concurrent analysis of product assembly and disassembly. *Proceedings of the 9th International Conference on Computer Supported Coorporative Work in Design*: 893-898.

Yu, K., K. Y. Goldberg. 1998. A complete algorithm for fixture loading. *International Journal of Robotics Research*. 17(11): 1214-24, November.

Zha, X. F., H. Du. 2002. A PDES-STEP based model and system for concurrent integrated design and assembly planning. *Computer Aided Design*, 34: 1087-1110.

Zhang, H., D. Manocha, T. Hudson, K.-E. Hoff III. 1997. Visibility culling using hierarchical occlusion maps. *Proceedings of ACM SIGGRAPH 97*: 77-88.

Zhang, L., Y. J. Kim, D. Manocha. 2007. A Hybrid Approach for Complete Motion Planning. *Proceedings of the 2007 IEEE/RSJ International Conference on Intelligent Robots and Systems*. San Diego, CA, USA, October 29-November 2.

Curriculum Vitae

EDUCATION

Doctor of Philosophy University of Manitoba, Winnipeg, Canada

Master of Science Beijing Institute of Machinery, Beijing, China

Bachelor of Engineering Tsinghua University, Beijing, China

PUBLICATIONS

Peer-reviewed Journal Papers

1. Kang, X., Q. Peng. 2010. A new algorithm for motion planning of fixture loading. *Transactions of the North American Manufacturing Research Institution of SME* 38: 695-702.

2. Kang, X., Q. Peng. 2010. Data integration from product design to assembly planning in a collaborative environment, *International Journal of Manufacturing Research* 5(1): 120-137.

3. Kang, X., Q. Peng. 2009. Recent Research on Computer-Aided Fixture Planning. *Recent Patents on Mechanical Engineering* 2(1): 8-18.

4. Duan, X., G. G. Wang, X. Kang, Q. Niu, G. Naterer, Q. Peng. 2009. Performance study of mode-pursuing sampling method. *Engineering Optimization* 41(1): 1-21.

5. Peng, Q.-J., X.-M. Kang, T.-T. Zhao. 2009. Effectiv*e VR-based building navigation using dynamic load*ing and path optimization. *International Journal of Automation and Computing* 6(4): 335-343. DOI: 10.1007/s11633-009-0335-9.

6. Kang, X., Q. Peng. 2008. Tool feasibility analysis for fixture assembly planning, *Journal of Manufacturing Science and Engineering, Transactions of the ASME* 130(4): 0410101-9.

7. Kang, X., Q. Peng. 2008. Fixture assembly planning in a Web-based collaborative environment. *International Journal of Internet Manufacturing and Service* 1(2): 176-193.

8. Kang, X., Q. Peng, Q. 2008. Fixture feasibility: methods and techniques for fixture planning. *Computer-aided design and applications* 5(1-4): 424-433.

9. Kang, X., Y. Zhu. 2003. Computer-aided Analysis for 3D Locating Error of Fixture, *Manufacturing Technology & Machine Tool* 11: 59-64 (in Chinese).

10. Kang, X., Y. Zhu, Y. 2002. Programming with Visual Basic for numerical control machining. *CAD/CAM Yu ZhiZaoYe XinXiHu,* 9: 62-65 (in Chinese).

11. Kang, X., Z. Yu. Y. Zhu. 2001. Application of macro in programming for NC turning. *Modern Manufacturing Engineering* 10(253): 22-23 (in Chinese).

12. Kang, X., Z. Yu, Y. Zhu. 2001. Overcut and undercut in NC milling curves in cam with a slot. *Modern Manufacturing Engineering* 9(252): 25-26 (in Chinese).

Peer-reviewed Conference Presentations and Papers

1. Kang, X., Q. Peng. 2008. Computer-aided fixture planning: a review, *Proceedings of the ASME 2008 Design Engineering Technical Conferences and Computers and Information in Engineering Conferences (IDETC/CIE2008)* 5: 209-218. August 3-6, New York City, NY, USA.

2. Kang, X., Q. Peng. 2008. Integration of CAD Models with Product Assembly Planning in a Web-based 3D Visualized Environment. *IDMME-Virtual Concept*. October 8-10, Beijing, China.

3. Kang, X., Q. Peng. 2007. Analysis of tool accessibility in fixture setup planning, Proceedings of the *ASME 2007 Design Engineering Technical Conferences and Computers and Information in Engineering Conferences (IDETC/CIE2007)* 4: 1007-16. September 4-7, Las Vegas, Nevada, USA.

4. Kang, X., Q. Peng. 2006. The improvement of VE performance using cell segmentation, *Proceedings of the 16th CIRP International Design Seminar*, July 16-19, Kananaskis, Alberta, Canada.

5. Kang, X., Q. Peng, G. Thomas, C. Yu. 2006. Blind image restoration using the cepstrum method. *IEEE 2006 Canadian Conference on Electrical and Computer Engineering:* 1952-1955. May 7-10, Ottawa, Canada.

HONOURS AND AWARDS

NSERC Postdoctoral Fellowship, *NSERC*, 2010

Graduate Student Travel Award, *University of Manitoba*, 2010

UMGSA Travel Award, *University of Manitoba*, 2010

Edward R. Toporeck Graduate Fellowship in Engineering, *University of Manitoba*, 2009

NSERC Postgraduate Scholarship, *NSERC*, 2008

University of Manitoba Graduate Fellowships (declined), *University of Manitoba*, 2008

Graduate Student Travel Award, *University of Manitoba*, 2008

UMGSA Travel Award, *University of Manitoba*, 2008

Graduate Student Travel Award, *University of Manitoba,* 2006

Berdie & Irvin Cohen Fellowship in Engineering, *University of Manitoba*, 2005

Youth five Small achievement foster award, *China Aerospace Industry Co., China*, 1996

First class Qiu Hedun Scholarship, *Beijing Institute of Machinery, China,* 1994

PROFESSIONAL AFFILIATIONS

Member of the American Society of Mechanical Engineers (ASME)

Member of the Canadian Society of Mechanical Engineers (CSME)