

Spring 2000

Virtual reality based creation of concept model designs for CAD systems

Bilal Yehya Maiteh

New Jersey Institute of Technology

Follow this and additional works at: <https://digitalcommons.njit.edu/dissertations>



Part of the [Mechanical Engineering Commons](#)

Recommended Citation

Maiteh, Bilal Yehya, "Virtual reality based creation of concept model designs for CAD systems" (2000). *Dissertations*. 410.
<https://digitalcommons.njit.edu/dissertations/410>

This Dissertation is brought to you for free and open access by the Theses and Dissertations at Digital Commons @ NJIT. It has been accepted for inclusion in Dissertations by an authorized administrator of Digital Commons @ NJIT. For more information, please contact digitalcommons@njit.edu.

Copyright Warning & Restrictions

The copyright law of the United States (Title 17, United States Code) governs the making of photocopies or other reproductions of copyrighted material.

Under certain conditions specified in the law, libraries and archives are authorized to furnish a photocopy or other reproduction. One of these specified conditions is that the photocopy or reproduction is not to be “used for any purpose other than private study, scholarship, or research.” If a user makes a request for, or later uses, a photocopy or reproduction for purposes in excess of “fair use” that user may be liable for copyright infringement,

This institution reserves the right to refuse to accept a copying order if, in its judgment, fulfillment of the order would involve violation of copyright law.

Please Note: The author retains the copyright while the New Jersey Institute of Technology reserves the right to distribute this thesis or dissertation

Printing note: If you do not wish to print this page, then select “Pages from: first page # to: last page #” on the print dialog screen

The Van Houten library has removed some of the personal information and all signatures from the approval page and biographical sketches of theses and dissertations in order to protect the identity of NJIT graduates and faculty.

ABSTRACT

VIRTUAL REALITY BASED CREATION OF CONCEPT MODEL DESIGNS FOR CAD SYSTEMS

**by
Bilal Yehya Maiteh**

This work introduces a novel method to overcome most of the drawbacks in traditional methods for creating design models. The main innovation is the use of virtual tools to simulate the natural physical environment in which freeform design models are created by experienced designers. Namely, the model is created in a virtual environment by carving a workpiece with tools that simulate NC milling cutters.

Algorithms have been developed to support the approach, in which the design model is created in a Virtual Reality (VR) environment and selection and manipulation of tools can be performed in the virtual space. The tool trajectories are generated by the designer's hand movements and they are obtained by recording the position and orientation of a hand mounted motion tracker. Swept volumes of virtual tools are generated from the geometry of the tool and its trajectories. Then Boolean operations are performed on the swept volumes and the initial virtual stock (workpiece) to create the design model.

Algorithms have been developed as a part of this work to integrate the VR environment with a commercial CAD/CAM system in order to demonstrate the practical applications of the research results. The integrated system provides a much more efficient and easy-to-implement process of freeform model creation than employed in current CAD/CAM software. It could prove to be the prototype for the next-generation CAD/CAM system.

**VIRTUAL REALITY BASED CREATION
OF CONCEPT MODEL DESIGNS FOR CAD SYSTEMS**

**by
Bilal Yehya Maiteh**

**A Dissertation
Submitted to the Faculty of
New Jersey Institute of Technology
in Partial Fulfillment of Requirements for the Degree of
Doctor of Philosophy in Mechanical Engineering**

Department of Mechanical Engineering

May 2000

Copyright © 2000 by Bilal Yehya Maiteh

ALL RIGHTS RESERVED

APPROVAL PAGE

VIRTUAL REALITY BASED CREATION OF CONCEPT MODEL DESIGNS FOR CAD SYSTEMS

Bilal Yehya Maiteh

Dr. Ming C. Leu, Dissertation Advisor Professor of Mechanical Engineering, NJIT	Date
--	------

Dr. Denis Blackmore, Dissertation Co-Advisor Professor of Mathematical Science, NJIT	Date
---	------

Dr. Michael Recce, Dissertation Co-Advisor Associate Professor of Computer and Information Science, NJIT	Date
---	------

Dr. Rajesh Dave, Committee Member Professor of Mechanical Engineering, NJIT	Date
--	------

Dr. Frank Shih, Committee Member Professor of Computer and Information Science, NJIT	Date
---	------

Dr. Zhiming Ji, Committee Member Associate Professor of Mechanical Engineering, NJIT	Date
---	------

BIOGRAPHICAL SKETCH

Author: Bilal Yehya Maiteh
Degree: Doctor of Philosophy in Mechanical Engineering
Date: May 2000

Undergraduate and Graduate Education:

- Doctor of Philosophy in Mechanical Engineering
New Jersey Institute Technology, Newark, NJ, USA, 2000
- Master of Science in Mechanical Engineering
Jordan University, Amman, Jordan, 1997
- Bachelor of Science in Mechanical Engineering
Jordan University, Amman, Jordan, 1995

Major: Mechanical Engineering

Publications:

- Maiteh, B.Y, Leu, M.C., Blackmore, D., and Abdel-Malek, L., Swept-volume computation for virtual reality application of machining simulation, Journal of Materials Processing and Manufacturing Science (submitted).
- Maiteh, B.Y, Leu, M.C., Blackmore, D., and Abdel-Malek, L., Swept-volume computation for virtual reality application of machining simulation, Proceedings of the ASME Symposium on Virtual Environments for Manufacturing, vol. 5, pp.1-9, 1999.
- Abdel-Malek, L., Wolf, C. and Maiteh, B.Y, Tele-manufacturing as a means for improving industrial productivity in the Middle East, Proceedings of the International 3rd Jordanian Mechanical and Industrial Engineering Conference, 793-804, 1999.

- Maiteh, B. Y., Jubran, B.A., Influence of mainstream flow history on film cooling and heat transfer from two rows of simple and compound angle holes in combination, International Journal of Heat and Fluid Flow, 20(2): 158-165, 1999.
- Jubran, B.A, Maiteh, B. Y., Film cooling and heat transfer from a combination of two rows of simple and/or compound angle holes in inline and/or staggered configuration," Journal of Heat and Mass Transfer, 34(6): 495-502, 1999
- Leu, M. C., Blackmore, D., and Maiteh, B. Y., Deformed swept volume analysis of NC machining simulation with cutter deflection, Proceedings of International Conference on Sculptured Surface Machining /IFIP TC5 WG5.3, 158-166, 1998.
- Maiteh, B.Y., Film cooling and heat transfer with air injection through two rows of compound angle holes, Masters Thesis, Jordan University, Amman , Jordan, May 1997.

This dissertation is dedicated to
my family for their love and support

ACKNOWLEDGMENT

The author would like to express his deepest thanks to his thesis advisor Dr. Ming C. Leu, for remarkable guidance, constant support and encouragement through his studies. The author would also like to express his special appreciation to Dr. Denis Blackmore from the Mathematical Sciences Department, his co-advisor and the chairman of his dissertation committee, for the resourceful and insightful advice and support during his research. The author's deepest gratitude and appreciation also goes to Dr. Michael Recce of the Computer and Information Sciences Department, his co-advisor, for his constant guidance, encouragement and patience.

The author would like to thank Dr. Zhiming Ji and Dr. Rajesh Dave of the Mechanical Engineering Department, and Dr. Frank Shih of the Computer and Information Sciences Department for their valuable suggestions to this dissertation and for serving as members of the committee.

The author is grateful to Dr. Kane and Dr. Chen for graduate advice and assistance through these years.

The author would like to acknowledge Dr. Zhang, Mr. David Jack and Miss Lianhua Fu and the other members of the research groups for their help and support.

And finally, the author would like to express his deepest gratitude to his family members, especially his parents, for their love, faith and patience and many sacrifices.

TABLE OF CONTENTS

Chapter	Page
1 INTRODUCTION	1
1.1 Motivation	1
1.2 Problem Statement	3
1.3 Objectives and Scope of the Research	4
1.4 Contributions	5
1.5 Overview of the Thesis Presentation	6
2 VIRTUAL REALITY AND GRAPHICS SIMULATION.....	8
2.1 Graphical Simulation.....	8
2.2 Virtual Manufacturing	10
2.3 Virtual Designing	12
2.4 Related Virtual Reality Applications	15
3 VIRTUAL CREATION SYSTEM FOR FREEFORM SURFACES AND OBJECTS	17
3.1 Introduction	17
3.2 System Overview	18
3.3 Virtual Reality Hardware Devices.....	19
3.4 Solid Modeler of Virtual Reality Environment	23
3.4.1 Geometric Model of Virtual Workpiece and Tool	24
3.4.2 Manipulating Virtual Workpiece and Tool	27
3.5 Operating Virtual Workpiece and Tool	32
3.6 Ease of Use	34

TABLE OF CONTENTS (Continued)

Chapter	Page
3.7 Summary of the Chapter	34
4 SOLID MODELING ENGINE OF THE VIRTUAL REALITY ENVIRONMENT	36
4.1 Introduction	36
4.2 Overview of Solid Modeling Techniques	37
4.3 Solid Modeling for Real-Time Application	41
4.4 Analysis, Representation and Implementation of Swept Volume by Sweep Differential Equation	43
4.4.1 Related Studies	43
4.4.2 Sweep Differential Equation	46
4.4.3 Development of Computer Program	49
4.5 Solid Modeling by Ray-Casting Technique	58
4.5.1 Related Studies	58
4.5.2 Basis of Ray-Casting	59
4.5.3 Ray-Casting Algorithm	62
4.5.4 Boolean Operations by Ray-Casting Technique	68
4.5.5 The Cost of Ray-Casting	69
4.5.6 Reduction of the Cost and Acceleration	71
4.6 Summary	80
5 REPRESENTATION OF THE CREATED DESIGN MODELS	81

TABLE OF CONTENTS (Continued)

Chapter	Page
5.1 Introduction	81
5.2 Integration of the VR Hardware with the Solid Modeling Software.....	82
5.2.1 Integration of the SDE Method with Ray-Casting Technique	82
5.2.2 Integration of Solid Modeling Engine with VR Hardware Environment	91
5.3 Integration of the VR System with the Commercial CAD/CAM Packages..	97
5.3.1 Overview of NURBS Representation for Curves and Surfaces	98
5.3.2 Conversion of Pixel Representation to NURBS Representation.....	100
5.4 Analysis of the Design Model Accuracy	106
5.4.1 Uncertainty and Accuracy of the VR Hardware	107
5.4.2 Accuracy of the Solid Modeling Engine	113
5.5 Demonstration Examples for the VR system	119
5.5.1 Example 1	119
5.5.2 Example 2	121
5.5.3 Example 3	123
6 INTERACTIVE BOOLEAN OPERATIONS FOR DESIGNING 3D SOLIDS IN A VIRTUAL REALITY ENVIRONMENT	126
6.1 Introduction	126
6.2 Background	126

TABLE OF CONTENTS

(Continued)

Chapter	Page
6.3 Interactive Boolean Operations Approach	132
6.4 The Interactive Boolean Operations and SDE Method for Virtual Reality	
Modeling of 3D Solids	134
6.4.1 The Delanuay Algorithm	136
6.4.2 Classifying the Points of the Object	139
6.5 The Complexity of the Integration Methods	140
7 CONCLUSIONS AND FUTURE WORK	142
7.1 Conclusions and Major Contributions.....	142
7.2 Future Work	143
REFERENCES	145

LIST OF FIGURES

Figure	Page
3.1 The virtual reality environment	19
3.2 A tracking device used to direct the motion of the virtual tool	21
3.3 The tracker motion device	22
3.4 Overall system architecture	24
3.5 Wire profile describing the cross-section of a solid	25
3.6 Solid created by sweeping a profile along a wire	26
3.7 Virtual reality control window	27
3.8 The tool sub menu for manipulating the tool geometry	28
3.9 Rotate option window for orienting the VR environments	29
3.10 The Move option menu	30
3.11 The Zoom Option window for examining the objects	31
3.12 The VR environment including virtual workpiece and tool	32
4.1 CSG tree example combines of two primitives	38
4.2 Sweeps. (a) 2D areas are used to define (b) translation sweeps	40
4.3 Partitioning the boundary of an object	47
4.4 Coordinate frame transformation	50
4.5 Virtual tool discretization	53
4.6 The connection of grazing points	57
4.7 Camera model and local coordinate system	59
4.8 Example of ray-casting	60
4.9 Data structure of ray-casting	62

LIST OF FIGURES (Continued)

Figure	Page
4.10 Local coordinate system of primitives: (a) The cylinder's axis lies along the Z-axis with one end at origin, (b) The sphere is centered at the origin	64
4.11 Check the intersection point for inclusion in the triangle: (a) all normal vectors in the same direction, (b) normal vector "N012" is not in the same direction as normal vectors "N023" and "N031"	67
4.12 Classification of Boolean operation cases	68
4.13 Traditional Bound box	72
4.14 The swept volume shape	73
4.15 The improved and traditional bounding boxes	74
4.16 The scan rendering technique	76
4.17 The traditional transformation mechanism in ray-casting	78
4.18 New transformation mechanism in ray-casting	79
5.1 The data structure of the swept volume	84
5.2 The data flow between SDE and ray-casting methods using the queue data structure	86
5.3 Integration of the SDE method with ray-casting technique	88
5.4 One-dimensional Boolean operation	90
5.5 The integration of hardware with software	92
5.6 Converting from the pixel representation to boundary representation	95
5.7 Cases of pixel point connections for constant x address	101

LIST OF FIGURES (Continued)

Figure	Page
5.8 Cases of pixel point connections for curves	102
5.9 The position tests for the motion tracker in: a) X-direction, b) Y-direction and c) Z-direction positions	109
5.9 The orientation test for the motion tracker: a) Azimuth, b) Elevation, and c) Roll angles	110
5.11 The measured Y-values from motion tracker vs. the ideal Y-values for (a) Trial one (b) Trial two	112
5.12 Comparison of the error percentage in Y-values in trials one and two	113
5.13 Approximation error of virtual tool in pixel representation	115
5.14 The tool motion	116
5.15 The feedback system to adjust the user hand speed	118
5.16 The NJIT Logo created by our VR system	119
5.17 The start step in designing the mouse	121
5.18 The second step in creating the freeform surface of the mouse	121
5.19 Final step in mouse design: (a) removing the extra material around design model (b) the final design model of the mouse	122
5.20 First cut of Mickey Mouse	123
5.21 The final stages of Mickey model (a) Removing the extra material with a larger tool (b) the final design model of Mickey	123
5.22 The B-spline curves of the Mickey face model after importing into CAD/CAM package	124
5.23 The solid model of the Mickey head in two different directions after being imported to Pro/Engineer as NURBS representation	125

LIST OF FIGURES **(Continued)**

Figure	Page
6.1 The ordinary Boolean intersections of two cubes may produce (a) a solid, (b) a plane, (c) a line, (d) a point, or (e) the null set	128
6.2 Steps of Boolean operation	131
6.3 Steps of the interactive Boolean operations method	133
6.4 Implementation of the Delaunay algorithm	137
6.5 The new triangulation after inserting the datum X	138
6.6 Classifying the grazing points	140

LIST OF TABLES

Table		Page
5.1	The depth values and y-addresses for Figure 5.8	104
5.2	The depth values and y-addresses for Figure 5.8.....	105
5.3	The technical specifications of the FOB.....	108

CHAPTER 1

INTRODUCTION

1.1 Motivation

Growing interest in virtual reality (VR) techniques over the past few years has lead to numerous applications, such as in landscaping, interior design, and video games. Engineers have been using computers for many years to create design models and generate Numerically Controlled (NC) machining programs, but the present CAD/CAM (Computer Aided Design/Computer Aided Manufacturing) systems lack the interactive capabilities available from a VR environment.

Numerically controlled machines such as NC milling machines are computer-controlled machines that move a cutting tool through a sequence of tool motions under the direction of a computer program. NC machines are widely used to machine complicated parts in the aerospace, automobile, mold/die making, and many other industries. However, it is often difficult to achieve high productivity with NC machines because of the usual errors in the generation NC machining trajectories for parts of complicated shapes.

The conventional process of design model creation and NC machining trajectory generation begins with work done by a stylist, who creates a rough sketch of the product to demonstrate the design idea. Since the existing CAD packages require training and skill on the part of the user to create a good design model, the rough sketches are typically sent to a draftsman to develop the design model using a CAD package. The communication between the stylist and the draftsman is typically through rough

sketches and verbal instructions. Some design 'intent' is inevitably lost in this transfer of information. This loss of information is usually rectified by repeated consultations between the stylist and the draftsman until a mutually agreeable solution is reached and a final concept design is created on the computer. The design model is viewed on a 2D screen and may be modified to fit the requirements. At this stage it is not certain that the design model can be manufactured. The stylist and the draftsman usually do not consider machining issues during the design process. In some industries like the automotive industry, a skilled group of experts will create a full size clay model for the computer design model (the CAD model) in order to evaluate the design model in three dimensions. The clay model of the design may be modified to add more freehand features and adjusted to be closer to the original design style idea. Special and expensive materials are often used to build the clay model in order to give it a smooth surface and make it easy to manipulate. After the clay model for the design is approved, it is scanned and input to a CAD package. The engineer may modify and adjust the scanned data and then input it into the CAD package to create the computer design model.

After this stage, a computer aided manufacturing (CAM) software package is used to generate the NC machining program. Considerable experience, skill and training in manufacturing are needed to use the existing CAM packages effectively. After the NC program is generated, simulation and verification of the NC program is needed to check the tolerance between the design model and the simulated machined model. If the CAM package is unable to generate the NC program for a given design model, then the model will be modified in the CAD package or in the clay model. If the design part has

complicated shapes, much skill and time is required to produce an acceptable model and the NC machining program.

Most important in the tool path generation by an automatic programming system are the following problems:

1. The problem of recognizing the regions where cutting is impossible in a designated tool approach direction.
2. The problem of collision between the machine (including the workpiece) and the portion of the tool not involved in cutting.
3. The problem of recognizing an inter-surface relation between a region to be cut (which may be comprised of several surfaces) and the cutting edge of the tool.

1.2 Problem Statement

This traditional method of concept design and NC program generation has several drawbacks. The first drawback is the repeated consultations that must occur between the stylist and the draftsman which are time consuming. The second drawback is that building clay models is time consuming especially since the clay model features are affected by the temperature and the moisture of the environment. The third drawback is that scanning the clay model and interpolating the scan data into existing CAD systems are time consuming. The fourth drawback is that the tool path generation will affect the design process and change some features on the design model. Also the existing CAM packages impose limitations on generation of tool paths for freeform surfaces. As a result there may not always be sufficient time for the stylist to experiment with a wide range of part geometries.

The CAD/CAM systems in existence today have very limited ability to allow the stylist to perform conceptual designs. This is because most of the existing CAD/CAM systems require exact geometric specifications including shapes and dimensions, even though these details are unnecessary and often unknown at the conceptual design stage. Another reason is that the existing CAD/CAM systems have a non-intuitive, mouse and keyboard-based, interface which can be quite difficult to use for design purposes without extensive training.

Conceptual design is rarely supported in current CAD/CAM packages [Smithers, 1989]. The reason why conceptual design is so difficult to support is that it imposes the largest demands for interactivity. Initially, designers need to experiment with a large number of potential designs. This navigation in design space must be done in a very intuitive and fast manner. The vocabulary by which designer intentions are expressed should be very high-level, and must be translated into a large number of low-level operations that could be difficult to compute efficiently.

1.3 Objectives and Scope of the Research

The goal of this research is to develop a novel method for creating design models in a VR environment which overcomes most of the drawbacks in the traditional method and incorporates a strategy for generating NC machining trajectories in the design process. This VR environment enables the user to create design models in the same manner as in a natural physical environment by an experienced designer. Further, NC machining trajectories can be generated from the design model creation process. The research

addresses the needs for and issues pertinent to the creation of this more natural and intuitive VR environment for CAD/CAM applications.

The specific thesis objectives are:

1. Develop a VR environment that can be used to study new ways of design model creation and NC machining trajectory generation.
2. Develop an efficient method that is capable of real-time simulation of material removal process in VR.
3. Analyze the tradeoff between representation accuracy and computation requirements in the VR based design model creation process.
4. Demonstrate the usefulness of the results of the work for solving problems in a practical setting.

1.4 Contributions

This research makes a number of contributions to the field of CAD/CAM and geometric modeling:

1. A novel approach that uses a virtual environment to create complex surfaces for CAD applications.
2. A new algorithm for Boolean subtractions that is specifically adapted to the use of the Sweep Differential Equation (SDE) method. It integrates the SDE method with ray-casting and graphics techniques to perform real-time Boolean subtractions for objects in the VR environment.
3. An enhanced understanding of the tradeoff between representation accuracy and computation requirements in the VR based design model creation process.

4. A VR system for creation of freeform surfaces that can be readily integrated with the existing commercial CAD systems.

1.5 Overview of the Thesis Presentation

In this chapter, we have introduced motivations for research in the area of CAD/CAM, defined the problem that is studied in this thesis, discussed the objectives and scope of the work, and outlined the thesis contributions.

In Chapter two we will present a detailed survey of the previous work that has influenced or informed the current research. This includes the research that has led to the current CAD/CAM technology and current research on VR for design model creation and evaluation.

We present our method for creating design models and freeform surfaces in a VR environment in Chapter three. This formal and systematic methodology is the basis for the specific research that will be presented in later chapters.

Chapter four is devoted to the description of an algorithm for implementing the SDE method together with ray-casting and graphics techniques for creating freeform surfaces and design models.

In Chapter five, an approach for selection of geometric resolution parameters in the virtual design model creation, which uses flat-end and ball-nose virtual tools with discrete representation of the workpiece and cutting edge geometry, is presented. Integration of our VR system with the CAD system by using the NURBS representation for the design model is described. An empirical measurement is performed to quantify how the positional uncertainty of motion tracker in the VR system affects the

representation accuracy of the design model. Illustrative examples are presented to demonstrate how the VR system actually works.

We present a new algorithm for Boolean subtractions specifically adapted to the use of the SDE method in Chapter six. The new algorithm is described and its computational complexity is analyzed.

Finally, we conclude in Chapter seven with a detailed account of the main contributions of this research and possibilities for future work in this area.

CHAPTER 2

VIRTUAL REALITY AND GRAPHICS SIMULATION

The research presented here has roots in several diverse fields, and is related to many previous results. In this chapter, we will briefly discuss prior work in related disciplines that has an overall bearing on this work. This includes research from the field of graphical simulation, related work in the areas of virtual manufacturing, and current efforts to develop a virtual environment for design.

2.1 Graphical Simulation

The aim of most computerized NC simulation software is to simulate the machining processes that takes place in NC machining. All the methods described here are developed for off-line applications where the time is not critical. These methods are based on Boolean set theory, where the basic workpiece forms the starting block and is represented by a set. The volume swept out by the tool at each line in the NC program is represented by a second set, and is removed from the workpiece using the set difference operator. These methods simulate the machining process using the graphics capability of the computer to display an up to date representation of the workpiece being machined.

Wang and Wang [Wang & Wang, 1986] created a 5-axis NC verification system using a full depth pixel representation of the part being machined. The swept volume of the tool is computed as a parametric boundary surface, from which they create a polyhedral model, and then use an extended Z-buffer to create a full depth pixel image. This pixel image is then subtracted from the displayed workpiece. By comparing the

Ray-casting techniques have also been used to create a pixel image [Leu, et al., 1986], with the advantage of being able to deal with most solids, not just polyhedral models. The problem with this method, however, is that the amount of the time required to create the data is large because the data is created pixel by pixel. Owing to the nature of ray-casting techniques, the user is confined to the views specified at the start of the verification, and further views require the rerunning of the whole system. Another problem experienced with these methods is that graphics resolution plays a large part in the representation of the workpiece. As only one view can be selected at a time that the selected view is not likely to contain the whole machined area; this implies that the display will lack details, making some verification difficult.

Van Hook [1986] utilized an extended Z-buffer data structure for his graphical verification. He uses a Z-buffer method to convert surface data into his dixel (depth element) structure, and stored Z values for both the nearest and farthest surfaces at each dixel. A graphics level Boolean difference operation then took place to update the model. As with the previous methods, only the specified views are verified, but this is of little consequence as this method is quick and rerunning is a feasible option.

The problem with these graphical simulation methods is that the verification of the program is a visual process carried out by the user, with the computer only providing a single representation of the machined part, and not providing any aids to the correction of the program. The user has to ensure that the selected view contains all the information required, as the software has to be rerun in order to create new views. Hsu and Yang [1993], however, produced a simulator with the ability to create new views of the part by post-processing the created solid. They used an isometric projection of a voxel based

solid model of the part. Their technique is only applicable to 3-axis milling operations, as a Z-map structure is used to retain the height of the model at X-Y coordinate positions. An array containing the Z-axis heights is used to store the model data to produce alternate views of the part.

2.2 Virtual Manufacturing

The term virtual manufacturing seems an apt description of a project combining virtual reality and manufacturing in the 1990's, but was first used in 1982 by McLean et al [1983] when they were looking at ways to extend Group Technology (GT) methods. This research moved GT away from processing components in a physical machining cell on the shop floor, and instead creating virtual cells, which could allocate components to machines in different positions on the shop floor using a time share basis.

This term has also been used more recently by Mills et. al. [1993] who investigated software for designing and simulating manufacturing cells for products designed on CAD packages or geometric modelers. Kimura [1994] coupled the use of knowledge-based design with the modeling of engineering objects and the simulation of manufacturing on computers. Virtual reality techniques require real-time interaction with the virtual environment, so virtual manufacturing in this context implies interaction with a manufacturing environment, not just simulation.

Barrus [1993] presented a Virtual Workshop containing a selection of manually operated machines: band saw, drill press, milling machine, radial arm saw, and table saw. Using the handles and locks on each of the machines the user is able to create components in various sizes from a selection of materials. The system uses the ACIS,

which is a geometric modeler from Spatial Technology, exploiting the internal roll back mechanism to perform multiple undo and redo of operations, but not providing any other editing mechanism. The system does not present the workshop as a whole environment but with only one machine visible at a time; the required machine being selected from a menu and the display updated to show that machine.

A UK research team at the University of Bath has developed an interactive (mouse, keyboard, and menus) virtual manufacturing environment. This system models a machine shop floor containing a three-axis numerical control machine and five-axis robot for painting. The user mounts a workpiece on the milling machine, chooses a tool and performs direct machining operations, such as axial movements or defined sequences, or loads a part-program from memory [Bayliss et al, 1997]. This software does not provide the users with the ability to create their own machines or interact with them in a natural manner. All machining operations are first defined using menus and then results are updated to the user on the 2D screen.

A virtual workshop for mechanical design was developed at Massachusetts Institute of Technology [Barrus and Woodie, 1996]. The goal was to create a virtual workshop that can represent traditional tools and machinery that work like in a real machine shop. The objective of this work was to generate an environment for designers to do conceptual work by using virtual traditional tools in a virtual workshop. However, these virtual workshops restricted the opportunities for designers to be creative because the tools are rather limited. Also, the designers need to have extensive experience in order to use the virtual and real workshops to create the design model.

A group at Washington State University has developed a system for the early design evaluation of an automobile. This system utilizes Pro/Engineer models that are brought directly into a virtual design environment. Once immersed in the virtual environment, a user can evaluate the design, evaluate alternate designs and conduct ergonomic studies using full human body tracking [Angster et al, 1994].

Deneb Robotics has commercially available software for manufacturing simulation, virtual milling, virtual spray painting, virtual arc welding and Tele-robotics. Most of the theses systems are compiled software tools where all work is done using Deneb's graphics user interface on the screen for setting up the manufacturing plant, etc. and all subsequent interaction is also done on the screen via the mouse and keyboard.

2.3 Virtual Designing

Several prototype systems for direct shape modeling have been presented. They are implemented on many levels from simple viewing where predefined geometry is loaded into the virtual environment to fully interactive CAD applications where product geometry is created in the virtual environment.

Perles and Vance [1999] presented a method for editing NURBS surfaces using an input device. Tools enable user control over multiple points on a surface for gross surface manipulation. Changes in the surface normal can also be controlled through direct interaction with a display normal vector. In this application, first a predefined geometry is output from existing CAD packages as a NURBS representation and then it is input to the VR system for editing the geometry. This VR system can not generate geometry with sharp edges and the design part may not be manufactured.

COVIRDS [Dani et al., 1999] is a VR solid modeling application created to facilitate conceptual design in a virtual environment. The program makes 3D sketches with shape elements accessible to engineers, designer, and others. Modeling operations are selected and quantified with voice input and positioned with a data glove. The command structure is patterned after verbal descriptions to make control intuitive. This application uses existing CAD packages as a solid modeling engines for supporting changing of the predefined geometry. Each time the operator gives a command to perform an operation in a virtual environment, the program translates the command to a solid modeling option, and then solid modeling software performs the operations and converts the new solid to a polygonal representation. After that the geometry model is imported to the virtual environment for displaying.

Furlong [1997] created a Bezier freeform deformation based sculpting program for use in a CAVE type virtual reality environment. Users push or pull individual points on the surface and bezier volume to provide spline-based continuity to polygonal surfaces. The surfaces can also be painted using the paint brush tool to create the desired color. The program is useful for sketching freeform shapes during the conceptual design stage. The single point interaction method does not provide enough control for the modifications of production CAD models. This project is used to modify a predefined geometry input form existing CAD packages.

A Virtual Clay Modeling system is described by Kameyama [1997], in which a user can directly manipulate the shape of a virtual object like a clay model and can produce its solid model data. The key component of its hardware is a special input device with a 3D position tracker and a tactile sensor. The tactile sensor, which has an array of

small pressure sensors in a square area, is for detecting the force exerted by a user. Conventionally, NURBS are used for representing freeform surfaces of the virtual clay model. There are several limitations of this modeling method. First, the system does not create a complete shape since the surface grid points can move only vertically. Second, the method can be applied only when the initial shape of an operation surface is almost flat since the shape of the tactile sensor sheet is flat and unchangeable.

At the University of Illinois, Chicago, a prototype virtual reality based computer-aided design system has been designed and implemented. The focus of this work is to allow a simplified method of designing complex geometrical shapes through the use of a virtual reality environment [Zetu et al, 1999]. This project presents a construction method for a virtual factory model without using CAD packages, rather it employs computer vision techniques. The basic idea of this system is to extract 3D models from camera images from knowledge of the exact camera locations when capturing images and then perform image processing on the images to detect the edges of the 3D models.

The work at the Georgia Institute of Technology is focused on the application of an architectural method [Bowman et al, 1999]. The main idea of this project is to create conceptual building designs, in which the user has the ability to modify buildings, add details, or create new designs, all while immersed in the virtual world. This application is a natural extension of architectural walkthrough application. Also the designer has the ability to inspect building designs and this leads to a better understanding of the architectural space. Users can create simple buildings in an interactive, intuitive manner, simply by choosing vertices on the ground, then adding the third dimension by specifying a height for each vertex. The walls and ceiling are created automatically by the program.

Once the basic structure is in place, users may experiment with different colors and textures, add furniture to the interior of the space, or change the roofline, for example. A 3D menu system, 3D widgets for object manipulation, dialog boxes, slider widgets, and tool palettes have been used as user interfaces.

2.4 Related Virtual Reality Applications

Fang [1998] of Iowa State University described a virtual lab for knowledge-based learning and skills training. This work addressed the importance of using VR technology in engineering education. The basic ideas of developing a virtual machining lab were presented by showing that the virtual interactive learning environment is superior to straightforward computer simulation. The main advantage of this system is that it provides a safe, flexible and cost-effective learning environment. Also it gives students the opportunity to operate various machines and study machining theories on a one-to-one base.

Creating this kind of virtual machining lab is very difficult and may be impossible with the existing CAD/CAM packages for several reasons. First, creating the solid model of the virtual machine is very difficult and it is expensive to create graphical displays of the model. Second simulating the kinematics and dynamics of the real machine has until now been a problem in the CAM field. Also these applications do not provide real-time interaction, instead simulation of the machine operation is predefined.

Deviprasad and Kesavadas [1999] described an interactive tool for validating assembly components in a virtual environment using finite-element simulation. A simple assembly of two parts was considered to explain the concept of validating assembly

components in a virtual environment. The assembly quality was evaluated using both theoretically as well as physically based virtual models. Though clearance ratios provide clear results the complexity increases with increase in the number of mating parts. The system is still in the preliminary phase of development.

Levoy's team at Stanford University used a custom-built scanner to collect the data [Christensen, 1999]. The scanner projects onto an object a 6-inch line of laser light. When viewed from an angle, the line reflects the curves of the surface it covers. By tracking some 200 to 300 points as the line sweeps across the surface, a computer records an extremely detailed 3-D image. This process must be repeated many times to cover the entire surface of a statue.

CHAPTER 3

VIRTUAL CREATION SYSTEM FOR FREEFORM SURFACES AND OBJECTS

3.1 Introduction

As was said in Chapter 1, the current practical technologies for creating freeform surfaces or objects have several drawbacks and in this work a novel method introduced that overcomes most of these drawbacks. Examining the current research approaches discussed in Chapter 2, it is seen that most of these approaches have attempted to integrate the existing CAD/CAM packages, without alteration, with the new virtual reality technology. These attempts to use CAD/CAM in a virtual reality environment all lead to real-time interaction problems that are essentially impossible to solve.

Our approach is different from the current research approaches and existing technologies in that combines three-dimensional virtual reality, swept volume generation, solid modeling and Boolean operations in a new way to form a complete design model. The virtual reality environment for creating freeform surfaces and objects and its components, which include hardware devices and software programs, is described in this chapter.

3.2 System Overview

This thesis presents a novel method for representing the ideas of a stylist or designer directly within a computer environment, thereby making it possible to create very complex freeform surfaces in a natural and easy way. The main innovation is the use of virtual tools to simulate the natural physical environment in which freeform design models are created by experienced designers. Namely, the model is created in a virtual environment by carving a workpiece with tools that simulate NC milling cutters.

In this approach, the design model is created by the designer in a VR environment in which selection and manipulation of tools can be performed in virtual space. The tool trajectories are generated by the designer's hand movements and they are obtained by recording the position and orientation of a motion tracker mounted on the user's hand. Swept volumes of virtual tools are generated from the geometry of the tool and its trajectories. Then Boolean operations are performed on the swept volumes and the initial virtual stock (workpiece) to create the design model. After being post-processed by computer software, the trajectories of the virtual tool during the design model creation can be used as the trajectories of real cutters in actual NC machining.

The VR environment for creating the freeform surfaces and objects which has been developed includes hardware devices to allow the user to interact with VR objects, employ swept volume computation and representation methods to describe moving objects in the space, and solid modeler software to perform the Boolean operations between the virtual objects.

3.3 Virtual Reality Hardware Devices

The VR environment has been designed, as much as possible, to be user configurable, allowing the designer to model a particular engineering environment, including as much or little information as desired. This environment is depicted in Figure 3.1, which shows the hardware structure of the VR environment for creating a design model. The main system runs on a Personal Computer (PC) with Microsoft NT. 4.0 operating system. The hardware for the VR system can be divided into categories of input devices and output devices.

The input devices are used to assist in the communication of the user's ideas into computer representation. The input device in this system is the tracking device.

Tracking devices are used in general for two purposes: (a) to represent a portion

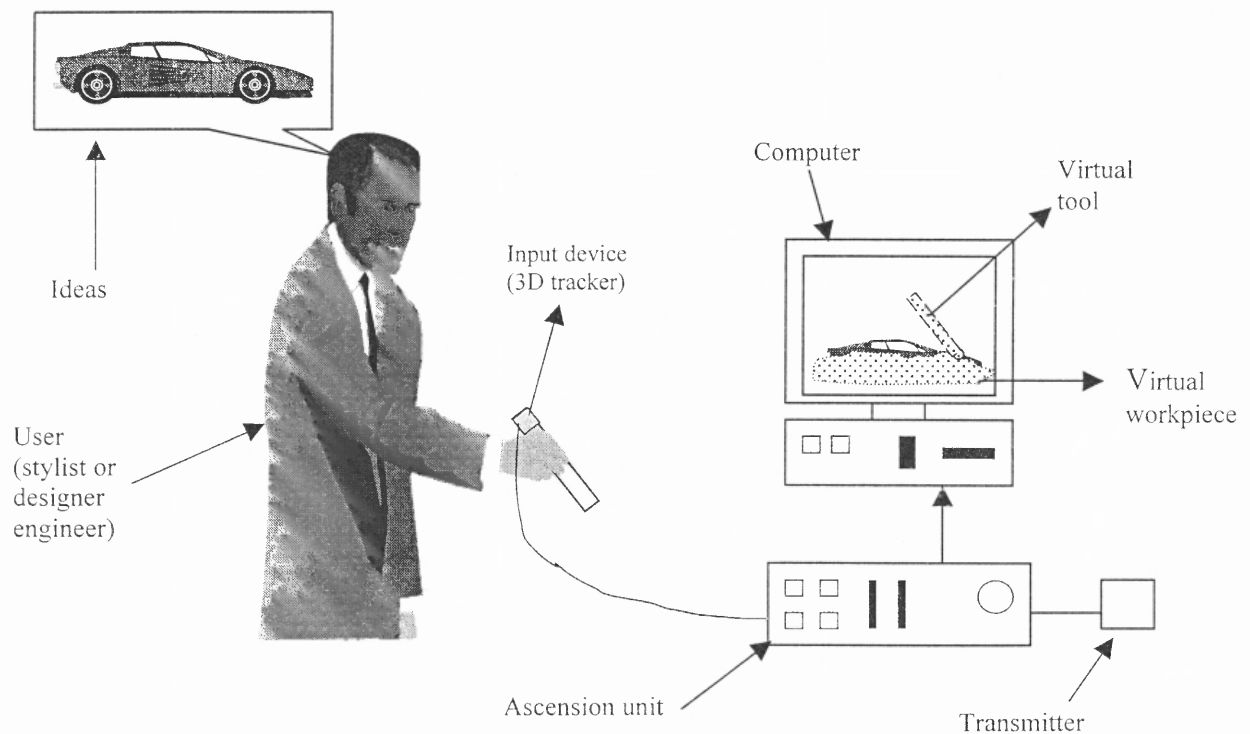


Figure 3.1 The virtual reality environment

of the user's body in the virtual environment, (b) to update the image display to the user. For representation, the trackers will be mounted within the pointing device or other convenient part of the user's body. The position and orientation data from the tracking device are then fed to the software to generate a replication of the user in the virtual environment. For display updates, the tracker is usually mounted on the user's head or tracks the eye direction, and uses this input information to display appropriate updated graphics. A tracking device typically uses one of the following technologies: ultrasonic, magnetic and mechanical.

In this work, the tracking device has been used to represent an NC milling machining tool in the virtual environment. Position and orientation data from the tracking devices and the geometry of the virtual tool are used to generate the virtual tool trajectories. The mounting of the motion tracker and its correspondence in the virtual world are depicted in Figure 3.2.

The position and orientation data from the tracker device which is mounted on the user's hand or gripped by the user represents the movement of the virtual tool in the virtual environment. These data describe the trajectories for the virtual tools. Since the tracker device has a limited workspace, a mapping between the tracker device and the virtual workspace has been built. The mapping relation is defined so that it compensates for the limitations of the screen size and tracker device workspace. In particular, the mapping is a one to one correspondence between the virtual spaces and actual tracker device workspace.

The tracker device that has been used in this work is a magnetic system. The magnetic system has acceptable accuracy (table 5.3) and performance in terms of the

virtual space volume. This system (as shown in Figure 3.3) consists of three distinct

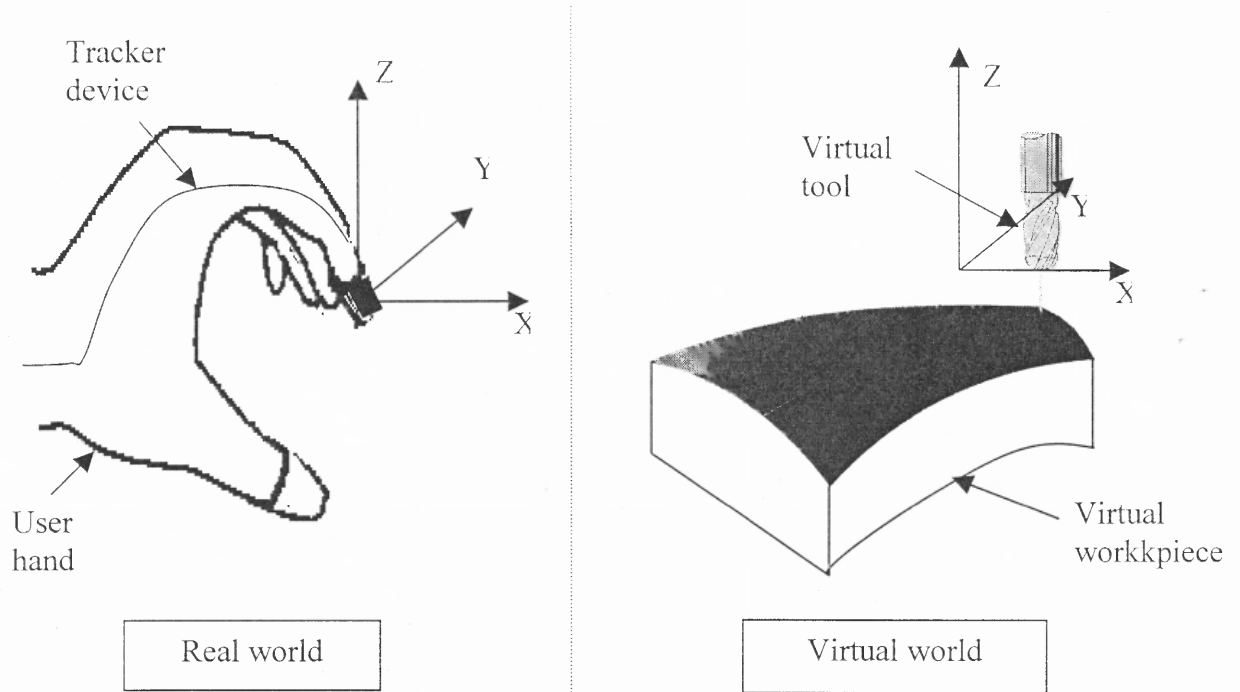


Figure 3.2 A tracking device used to direct the motion of the virtual tool

components: a transmitter to emit a pulsed DC magnetic field; a receiver to track the position and orientation; and an ascension unit to communicate sensed information to the host computer.

There are many output devices that can be used to communicate information generated by the computer software. These devices in general include several types of elements; for example, visual output devices, haptic devices (force feedback device) and auditory output devices. In our system we have integrated and implemented two output devices; namely visual output and auditory output devices.

The main objective of the visualization device is generally to display graphically the output of the geometric modeling computer software as a three-dimensional object in

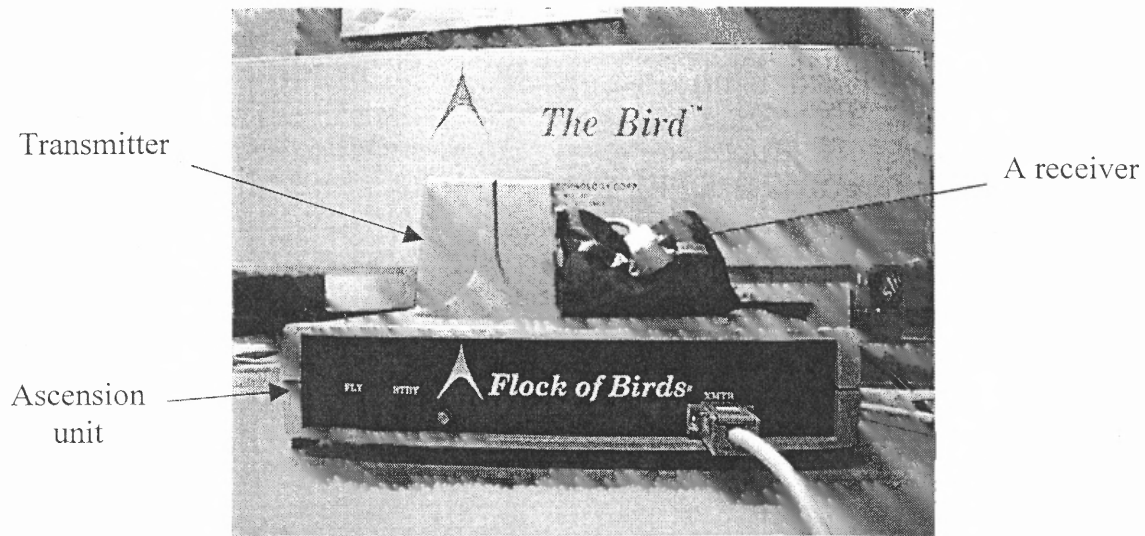


Figure 3.3 The tracker motion device

a VR environment. There are several classes of graphical display systems available, which include Head Mounted Display devices (HMD's), multi-wall displays (CAVE's), and the use of shutter glasses in combination with traditional CRT displays.

Shutter glasses are used in our VR system for creation of the freeform surfaces. Shutter glasses are one of the most feasible approaches for producing a "3D" viewing system. Shutter glasses are used to provide a three-dimensional image to the operator by sequentially blocking one eye's view of two images displayed on a traditional monitor. The images are sequentially displayed in synchronization with the shutter glass lens closing such that the left and right eyes are each presented a different view of the images, thereby creating a three-dimensional interpretation of the object. Shutter glasses are currently available for VR at low to moderate-low cost. The resolution of the image is limited by the monitor used and not the glasses, making this approach a good candidate

for our VR system. A high refresh rate monitor and glasses should be used to minimize the flicker effect, since each eye will be presented an image at one half the normal monitor refresh rate. The prime disadvantage of a shutter glasses and traditional monitor approach is that the images are confined to a relatively small portion of the user's field of vision, resulting in a low immersion effect.

In our actual implementation of the VR based system for the creation of freeform surfaces and objects, we used the traditional monitor to display the images of the virtual tool and virtual workpiece because of limitations of time and equipment.

The second device in the output system is the audio output device. To achieve a presentation of three-dimensional sound effects, a sound card is used. In this project, the audio output device is used to simulate material removal sound and the collisions between the user body and the other objects. The project relies on sound effects to provide realistic constraints.

An audio system has been integrated with our VR environment to give indications or feedback to the user if the virtual tool is cutting material from the workpiece or not. It is implemented by simply running on one of the buffers in the computer if the Boolean operation has begun. The music will terminate when the Boolean operation is not used.

3.4 Solid Modeler of Virtual Reality Environment

The second main component of the VR environment is the solid modeler for swept volume computation and representation in order to describe moving objects in the space, and an associated engine to perform the Boolean operations between the virtual objects. The overall system architecture is given in Figure 3.4.

This solid modeler contains five components, which are swept volume computation, Boolean operation, geometric model of a virtual workpiece and tool, display system and device and user interface. These components have been integrated in a novel way to enable the user or operator to communicate ideas directly into the computer system by creating a curvilinear object from the virtual workpiece using virtual tools.

The first two components (swept volume computation, Boolean operation) of the solid modeler are described in much greater detail in the next chapter. While, the other three components (geometric model of a virtual workpiece and tool, display system and device and user interface) are described in the following paragraphs.

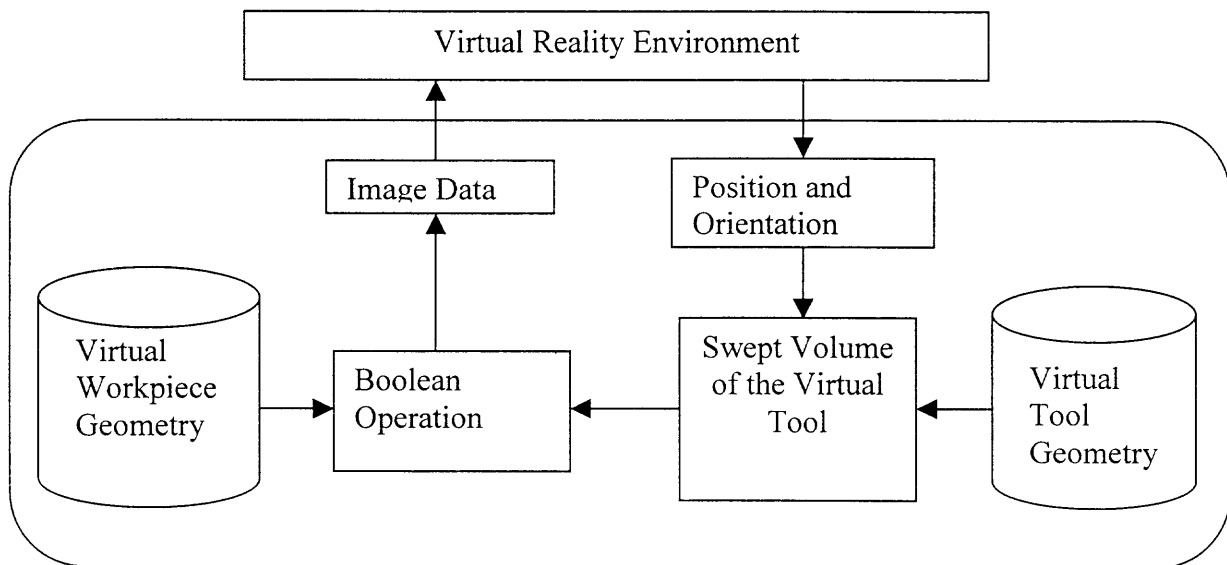


Figure 3.4 Overall system architecture

3.4.1 Geometric Model of Virtual Workpiece and Tool

Various representation techniques have been designed and developed to create solid models of real objects. The two most popular techniques are Constructive Solid

Geometry (CSG) and the Boundary Representation (B-rep). Our solid modeler system is based on boundary representation techniques [Zeid, 1991].

Boundary representation techniques, unlike CSG, store the model's boundary in the form of faces, edges and vertices. To ensure the topological consistency (no extra or omitted faces, edges or vertices) of the model, Euler's formula must remain invariant. Equation (3.1) is a simple example of this rule for polyhedral objects without holes

$$F + V = E + 2 \quad (3.1)$$

where F , V and E represent the number of faces, vertices and edges of the boundary of the solid object, respectively. The use of faces, edges and vertices allows B-rep modelers

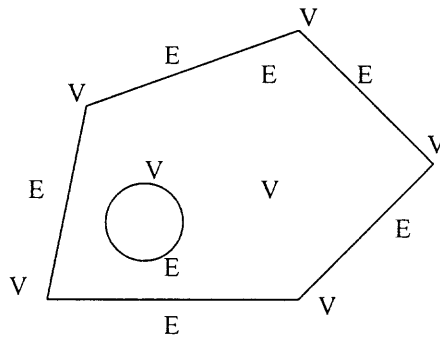


Figure 3.5 Wire profile describing the cross-section of a solid

to create models by sweeping profiles along wires. The cross-section (profile) of the desired solid is created from edges and vertices to form closed loops, which are converted into connected wires, as shown in Figure 3.5.

This cross-section can then be covered to form a sheet with two faces, which can be swept along a wire. The resulting body is composed of the two faces of the sheet solid connected at vertices by inserted edges which follow the path of the wire as shown in Figure 3.6.

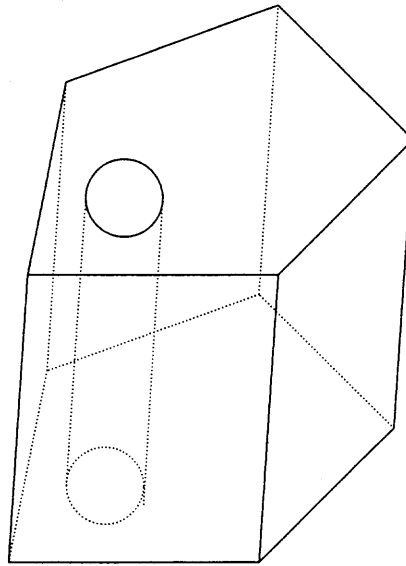


Figure 3.6 Solid created by sweeping a profile along a wire

Although B-rep modelers do not require the primitives provided by CSG modelers these common basic shapes are often included.

The virtual tools implemented in the virtual environment for creating freeform surfaces and objects are simulated NC milling machine tools. The set of tools modeled for mills are all cylindrical: flat-ended (slot-and end mills) and ball-ended with the ability to travel in six degrees (6 DOF) of freedom for linear interpolation. The flat-ended tool is represented by two circles with given radius and length.

Attributes have been enhanced for the virtual tool to restrict the virtual tool motion. These attributes are employed by restricting the virtual tool motion in one or

more degrees freedom as it travels in the VR environment. For example, if the operator or designer decides to move along a straight line in the direction of the x-axis. This can be done by attaching the x-axis attribute to the virtual tool through the software. The x-axis attribute or command simply means that the y, z coordinates of the position of the tracker device are ignored. In other words, the attributes of the virtual tools are simply means of ignoring one or more sets of data from the tracker device data (three positions and three orientations).

3.4.2 Manipulating Virtual Workpiece and Tool

Selecting interactive objects within the VR environment, such as workpieces and tools, leads to the creation of a control panel for the selected object; this can also be achieved by clicking the right mouse button within the VR environment window. An example of the control panel associated with a workpiece or tool can be seen in Figure 3.7. The

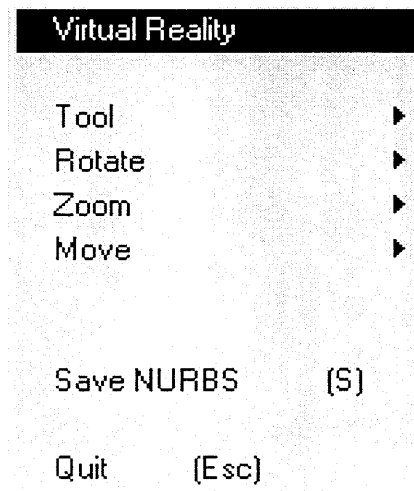


Figure 3.7 Virtual reality control window

control menu has five options in order to give the user sufficient flexibility for manipulating the workpiece, tool and VR environment. All of these options can be

activated by using the right button of the mouse and the left mouse button to employ the command.

The tool option allows the user to manipulate the virtual tool geometry. Selecting the tool option from the VR control window will activate the tool submenu as shown in Figure 3.8. The two basic elements in our tool geometry are the radius of the tool and the length of the tool, because shapes of all the virtual tools used in our implementation of the general method are cylindrical. That does not mean that our method for the creation of freeform surfaces and objects based on a virtual reality environment is limited to these types of virtual tools. The reason for using cylindrical (flat-end) tools is that they have a

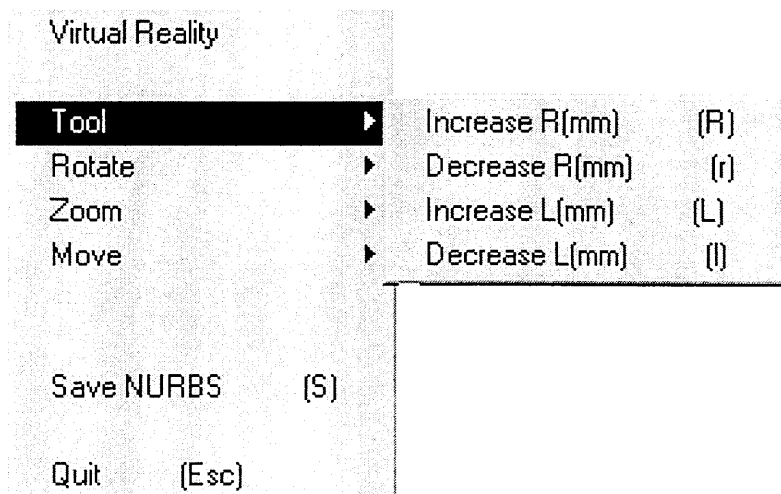


Figure 3.8 The tool sub menu for manipulating the tool geometry

simple geometry, yet they are versatile enough to create very complicated freeform surfaces and objects, It is not difficult to see that our method can readily accommodate more complex virtual tools within the VR environment.

Manipulation of the virtual tool geometry can be performed any time during the process of creating the design model, thereby providing the user great flexibility for

producing complex shapes. This flexibility is implemented by changing the radius and length of tool incrementally. Also the user has the additional option of increasing or decreasing the radius and length of the tool by using the keyboard. For example, pressing the key "R" on the keyboard will decrease the tool radius and using the combination of the keys "Shift" and "R" together will increase the radius of the tool. The same thing has been done for the length of virtual tool where the key "L" is used to decrease the tool length and the keys "Shift" and "L" together are used to increase the tool length.

The Rotate option can be used to orient the virtual workpiece in different directions. The user can rotate the virtual workpiece relative to the global three-axis coordinate system, comprised of the origin and x-, y- and z-axes. The purpose of the Rotate option is to enable the user to position the virtual workpiece in the virtual environment where it can most easily be machined to produce the curved design model.

This Rotate option (Figure 3.9) is needed in our current implementation in VR for creating the design model because the user has one screen as an interactive display,

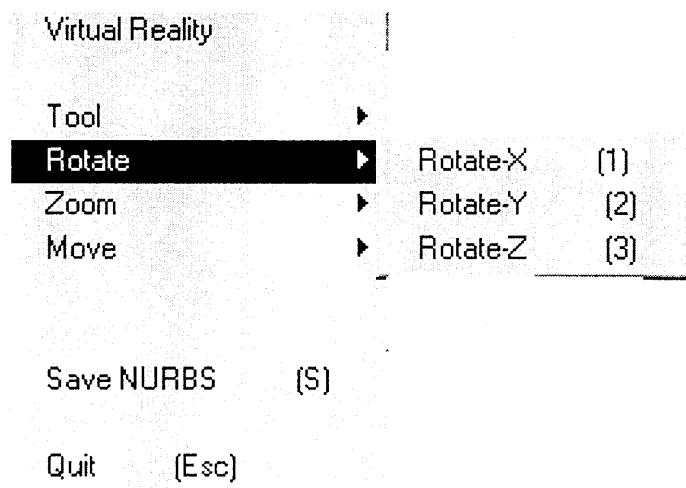


Figure 3.9 Rotate option window for orienting the VR environments

which means only a two-dimensional view is available. On the other hand, if the Head Mounted display is used for visualizing the virtual environment then this option still will be needed but will be implemented by the user in a different way. This alternate means visualizing the virtual environment is achieved by constructing a relationship between the orientation data from the motion tracker device, which is located on the Head Mounted Display device or any other visualization device, and the Rotate option.

The Move option has great impact on the navigation of the user around the virtual environment. This option allows the user to change the position of the workpiece relative to the global coordinate frame in two directions which are chosen to be the x-axis and the y-axis as seen in Figure 3.10.

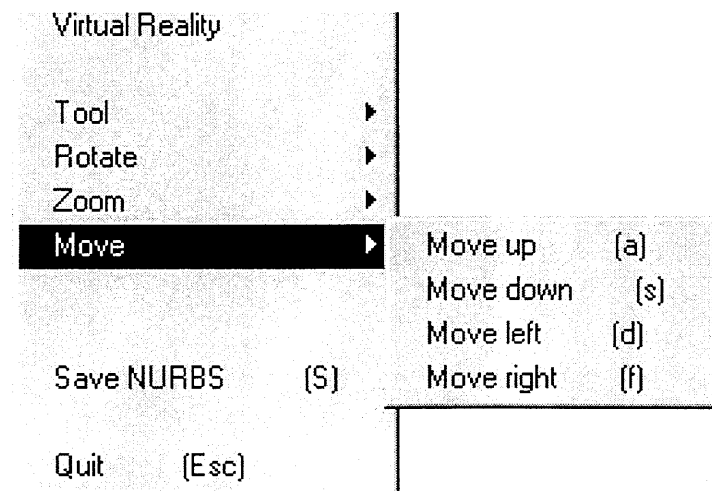


Figure 3.10 The Move option menu

The option Move Up is for moving the virtual workpiece in the positive y-axis direction and the option Move Down is for moving the virtual object in the negative y-direction. In addition, the options Move left and Move right allow the user to move

virtual objects along the x-axis. It is clear that the Move option is necessary in the virtual environment to enable the user to navigate around the workpiece.

The last option for manipulating the virtual environment and its objects is the Zoom option. The Zoom option can be used to examine the virtual world more closely. Each of the objects in the virtual world can be examined separately by selecting the examine option from the main viewer Pick pull-down Zoom submenu (Figure 3.11). By selecting a workpiece when it is on a tool, the designer can closely inspect the action of the tool during the process of designing any particular component, a much safer and less

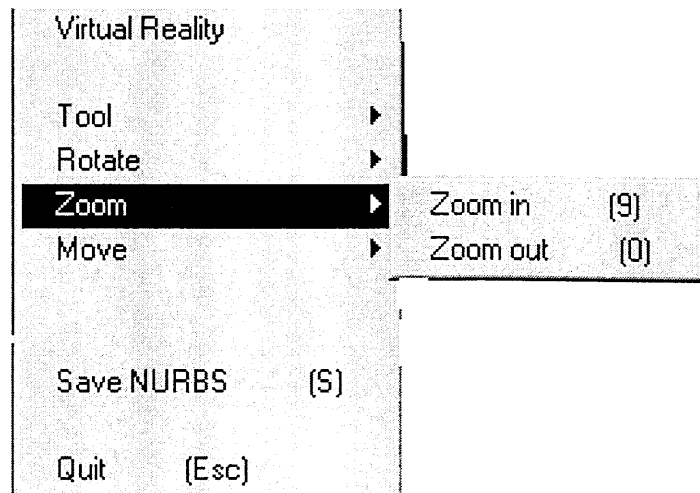


Figure 3.11 The Zoom Option window for examining the objects

costly option than a similar inspection in the real world.

The last two options in the control window of the VR environments are Save NURBS and Quit. The Save NURBS option will allow the user to save the design model in digital format in a file on the hard drive, if the user wants to stop the designing process and continue later. Within this option the design model will be reloaded automatically when the program starts again. Also this option can be used to export the design model in

NURBS form so that it can be integrated with existing CAD/CAM packages. The details of this option are presented in Chapter five. The Quit option permits the user to exit from the virtual environment without saving the results.

3.5 Operating Virtual Workpiece and Tool

When the designer starts the VR environment for creating a design model or freeform surfaces, a window is launched which presents the virtual workpiece and virtual tool as shown in Figure 3.12.

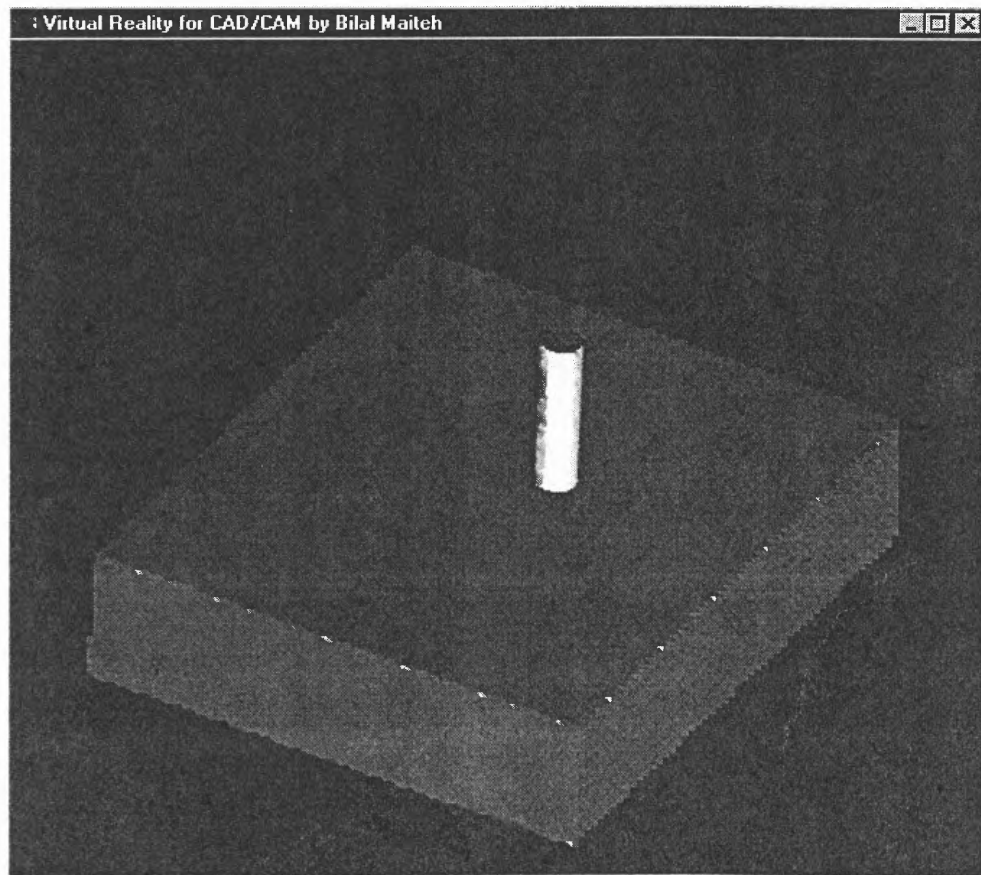


Figure 3.12 The VR environment including virtual workpiece and tool

The VR environment will enable the user to do the following:

1. Initialize the three-dimensional virtual solid workpiece by using the input system.
2. Create, select, and grip the virtual objects, which simulate the real machine tools.
3. Sweep the virtual objects (tools) in virtual space continuously to model the virtual workpiece geometry by subtracting the virtual swept volume (of the machine tool) from the current virtual workpiece.

The processing flow of design model creation and virtual tool trajectory generation in the virtual reality environment is as follows:

1. Create the virtual environment using the geometry modeler.
2. Display the three-dimensional virtual environment for the user via the shutter glasses.
3. The user selects the virtual workpiece, which is created by the geometry modeler, then places the virtual workpiece in the virtual environment with the input device.
4. The user selects and grips the virtual tool using the input device.
5. As the user sweeps the virtual tool in the virtual environment to create the design model, the tracking device records the position and orientation of the virtual tool.
6. By combining the geometry of the virtual tools and the position and orientation of the virtual tool data from the tracker device, the virtual tool paths are generated.
7. The tool paths, geometric model of the virtual workpiece and the virtual tool geometry are used as input for the design model creation method.
8. The geometry modeler (which supports the Boolean operations) software generates virtual machine tool swept volumes. It also performs Boolean operations between the virtual machine tool swept volumes and the virtual workpiece. Moreover, it simulates

the geometric changes of the virtual workpiece corresponding to a movement of the virtual tool. Actually, the changes in the virtual workpiece process represent the geometric design creation process.

9. The shutter glasses device updates the three-dimensional images of the changing virtual workpiece.

3.6 Ease of Use

The Virtual Creation System can be used by a varied selection of 'guinea pigs' ranging from young school children to captains of industry to the retired and elderly. After a quick introduction to the basics of the system, these users should be able to create their own designs on our virtual modeling, and then watch them exported to the existing CAD/CAM packages. Everyone who uses the Virtual Creation System is an expert within minutes, with the added satisfaction of taking their finished product away with them once it has been actually made.

3.7 Summary of the Chapter

The work described in this chapter has led to fundamental methods and algorithms useful to the development of VR based CAD/CAM systems. In order to emphasize the practical importance of these methods, our efforts are directed to the development of modules (including hardware and software) that can be integrated with commercial CAD/CAM systems.

VR hardware devices including the shutter glasses device, input device and motion tracker device are integrated with computer software developed to form the VR environment. A geometry modeler is used to develop computer models that represent

virtual tools and workpieces, whose images will be viewed by the shutter glasses. The movements of the tool are obtained by recording the position and orientation of the motion tracker. The tool used to remove material in the design model creation process is limited to NC milling cutters in order to insure that the created design model can be obtained by real NC machining. This constitutes the VR environment that is used for this research on a new way of design model generation and tool trajectory generation.

CHAPTER 4

SOLID MODELING ENGINE OF THE VIRTUAL REALITY ENVIRONMENT

4.1 Introduction

The Virtual Reality system for the creation of freeform surfaces and objects that will be described in the following chapters combines the virtual environment associated with all VR systems with the ability to interactively change the shape of objects within the environment in real time. This is achieved with the use of an underlying geometric modeler, which can produce a polygonal representation and display object being designed at any stage of the rendering process, in addition to storing the data corresponding to an approximated internal representation.

Within the constraints of the virtual tools represented in the virtual world, there are a finite number of kinds of changes in shape that each type of virtual tool can produce. So by supplying a group of modeling functions for each type, all actions possible in the real world can be recreated virtually.

An overview of the existing the geometrical modeler is presented in the beginning of this chapter. Then a solid modeler for creation of freeform surfaces and objects is presented with all of its components. The implementation of each component of the solid modeler is also described in detail.

4.2 Overview of Solid Modeling Techniques

Solid modeling is important in both CAD/CAM and computer graphics. For example in CAD/CAM, if a solid object can be modeled in a way that adequately captures its geometry, then a variety of useful operations can be performed before the object is manufactured. We may wish to determine whether two objects interfere with each other; for example, whether a robot will bump into objects in its environment, or whether a cutting tool will cut only the material it is intended to remove. In addition, some graphical techniques, such as modeling refractive transparency, depend on being able to determine where a beam of light enters and exits a solid object. These applications are all examples of solid modeling. The need to model objects as solids has resulted in the development of a variety of specialized ways to represent them. This section provides a brief introduction to most known representations, which are Constructive Solid Modeling (CSG), Boundary Representation (B-rep), Sweep Representation and Spatial-Partitioning Representations.

In Constructive Solid Modeling (CSG), simple primitives are combined by means of regularized Boolean set operators (intersection, union and difference) that are included directly in the representation. An object is stored as a tree with operators at the internal nodes and simple primitives at the leaves as shown in Figure 4.1. Some nodes represent Boolean operators whereas others perform translation, rotation, and scaling transformations. Since Boolean operations are not, in general, commutative, the edges of the tree are ordered.

CSG solid modeling is a very powerful representation method. It is easy to construct out of primitives and Boolean operations. It is concise and requires minimum

storage to maintain the necessary inventory of solid definitions (The CSG tree). This is why it is slow to retrieve the model because it has to build a boundary from the CSG tree.

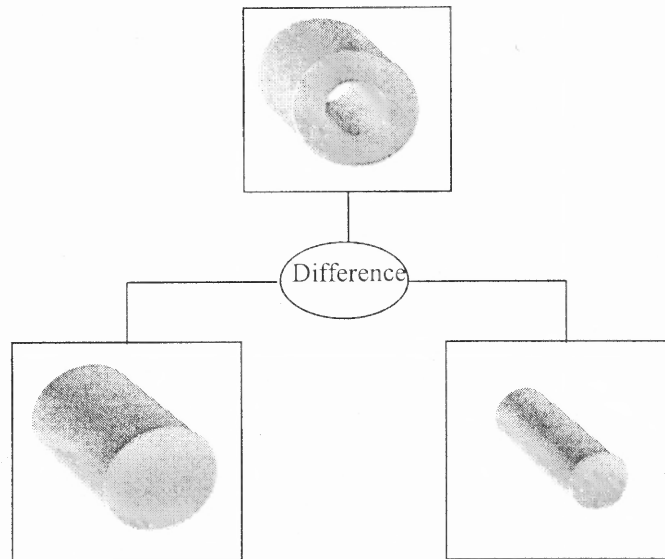


Figure 4.1 CSG tree example combines of two primitives

Another consequence of this low retrieval rate is that CSG is slow in generating wire frames, that is, line drawings. CSG must be converted internally into a B-rep to display the model or generate its line drawings. Application algorithms based on the CSG method are very reliable and competitive with those based on the B-rep method. However, the major disadvantage of CSG is its inability to represent sculptured surfaces and half-spaces. This is an active area of research.

Boundary representations (also known as B-rep) resemble the basic representations that we discussed in chapter three in that they describe an object in terms of its surface boundaries: vertices, edges, and faces as shown in Figure 3.6. Some B-reps are restricted to planar, polygonal boundaries, and may even require faces to be convex polygons or triangles. B-reps grew out of the simple vector representation used in earlier

research in computer aided geometric design and are used in many current modeling systems. Because of their prevalence in graphics, a number of efficient techniques have been developed to create smooth shaded pictures of polygonal objects.

The main advantage of the B-rep is that it is particularly well suited for the construction of solid models of unusual shapes that are difficult to build using primitives. Another major advantage is that it is relatively simple to convert a B-rep model into a wire frame model because the model's boundary definition is similar to the wire frame definition. One of the major disadvantages of the boundary model is that it requires large amounts of storage because it stores the explicit definition of the model boundaries. The model is defined in terms of simple boundary components such as edges and vertices whose number tends to grow rather rapidly if one requires accurate representations of complex models.

In sweep representation, sweeping an object along a trajectory through space defines a new object, called a swept volume. The simplest kind of sweep is defined by a two-dimensional (2D) area swept along a linear path normal to the plane of the area to create a volume. This is known as a translational sweep or extrusion and it is a natural way to represent objects made by extruding metal or plastic through a die with the desired cross-section as shown in Figure 4.2. Rotational sweeps are defined by rotating an area about an axis.

Sweeping operations are useful in engineering applications that involve swept volumes in space. Two widely used applications are simulations of material removal due to machining operations and interference detection of a moving object in space. In the first application, the volume swept by a moving cutter along a specific direction is

intersected with the raw stock of the part. The intersection volume represents the material removed from the part. In interference detection, a moving object collides with a fixed one if the swept volume due the motion of the first intersects the fixed object.

General sweeps are particularly difficult to model efficiently. For example, the trajectory and object shape may make the swept object intersect itself, making volume calculations complicated. In general, it is difficult to apply regularized Boolean set

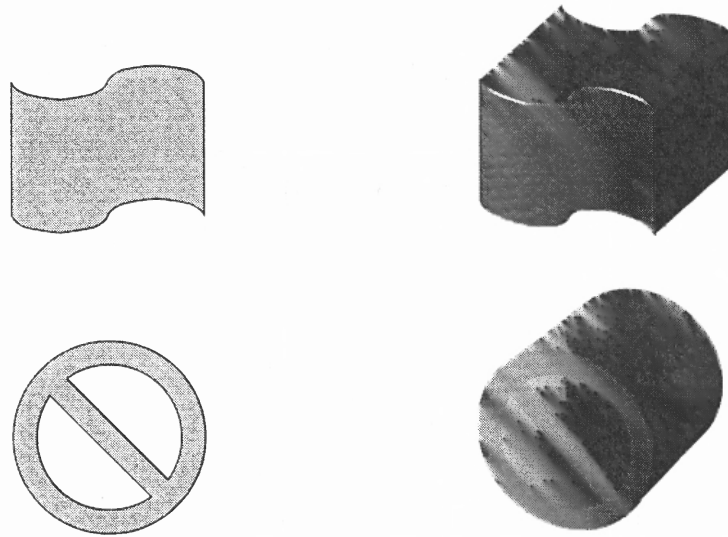


Figure 4.2 Sweeps. (a) 2D areas are used to define (b) translation sweeps

operations to sweeps without first converting to some other representation.

The last representation is the Spatial-Partitioning representation. In the Spatial-Partitioning representation, a solid is decomposed into a collection of adjoining, nonintersecting solids that are more primitive than, although not necessarily of the same type as, the original solid. Primitives may vary in type, size, position, parameterization, and orientation, much like the variously shaped pieces in a child's block set. The extent to which one decomposes objects depends on how small the primitive solids must be in order to efficiently and accurately perform the operations of interest.

The most popular method in Spatial-Partitioning representation is the spatial enumeration technique. In spatial enumeration, a solid is represented by a sum of spatial cells that it occupies. The cells (sometimes called voxels for "volume elements") are cubes of a fixed size that lie in a fixed spatial grid. Each cell can be represented by its centroid coordinates in the grid. The smaller the size of the cube, the more accurate the technique in representing curved objects. It is exact for box like object. The technique is unambiguous, essentially unique, and easy to validate, but it is verbose when describing an object, especially if it has significant curvature [Zeid, 1991].

4.3 Solid Modeling for Real-Time Application

To create the design model in our VR environment, the designer will manipulate the cutting tool to remove material from the stock. The workpiece geometry is constantly changing during this process. The updated workpiece model can be realized by performing Boolean subtraction of the tool swept volume from the workpiece. The design model has been created when this process is finished.

In order to build our VR environment for creation of design models, a solid modeler must be selected from the existing solid modeling techniques that have been introduced in the previous section or one has to create a new solid modeler to interactively change the shape of objects within the environment in real time. It is clear from the our description of the new approach for creating a design model that it is based on two main components. The first component is the instrumentation that allows the user to sweep a virtual tool in the virtual world in arbitrary motion without any restriction. Also, the shape of the virtual object can vary from a simple geometry to a complex shape.

The second component is the facility for removing material from the workpiece to create the design model in the VR environment. This must be performed in real time if the design process is to be truly useful. In terms of our VR based method for design, real time can be equated with a rate of the VR updates that enables the user to perform the virtual design process in a natural way without undue hesitation caused by relatively long waits for system updates.

A thorough examination of current solid modeling modeler system, shows that they are incapable of providing the real-time processing required for our system [Smithers, 1989]. The main cause of this inadequacy of existing solid modeler is they require inordinate amounts of computation time to compute and represent swept volumes with the level of accuracy required in our VR system. Not only that, but the Boolean operation also must be done fast enough to achieve the real time updating of the results for the user. In conclusion then, we find that we must develop a new solid modeling system for use in our VR based design process.

Our design model creation method uses a combination of the Sweep Differential Equation (SDE) method [Blackmore and Leu, 1990] for modeling swept volumes and the ray-casting method [Goldstein and Nagel, 1971] for Boolean operations. The swept volume of the tool will be constructed from the tool geometry and tool trajectory data, which represent the finger (mounted with the motion tracker) movements of the designer and are obtained by recording the position and orientation of the motion tracker. The key concern here is that the process of removing material from the workpiece to create the design model in the VR environment has to be performed in real time. This is more challenging than our previous work on NC machining simulation and verification, which

has been done off-line. We solved this problem by developing a new algorithm for Boolean subtraction, specifically adapted to the use of the SDE and ray-casting that is much more efficient than existing algorithms. The computational complexity of the algorithm is analyzed in Chapter 6.

4.4 Analysis, Representation and Implementation of Swept Volume by SDE

4.4.1 Related Studies

A great amount of effort has been devoted to the development of fast and accurate methods to represent and analyze the swept volume of a three-dimensional object under a general rigid-body motion. That effort was initiated in the late 1970's and 1980's to study manufacturing automation strategies. The most useful methods include the envelope theory [Wang and Wang, 1986a, 1986b], Z-buffer method [Narvekar, 1991; Sambandan, 1988, 1990], ray-casting method, or a combination of these methods [Leu et al., 1986; Weld and leu, 1990]

The envelope technique [Wang and Wang, 1986a, 1986b] was one of the earliest attempts to compute the swept volume generated by surfaces. Based on this theory, Sambandan [1988] developed a five-axis NC simulator for flat-end, ball-end and fillet-end cutters in image space and derived the parametric representation of the bounding surfaces of the cutter swept volume. The object's geometry also included any regular polyhedral objects [Sambandan, 1990]. Narvekar [1991] derived the envelope equations for cylindrical, conical, spherical and toroidal surfaces of a general 7-parameter APT (automatically programmed tools) tool and also conducted intersection calculations. These representations were combined to model the swept surfaces of standard NC milling tools

under general five-axis motions. Tool swept volumes were represented by these swept surfaces combined with the models of the tool in the initial and final positions.

Sambandan's and Narvekar's works rely on the hidden line removal method [Foley et. al., 1997] in computer graphics to find graphically the exact boundary of a swept volume. There are some other useful methods for representing and analyzing swept volumes. For example, fast and accurate renderings of general swept volumes may be obtained by using the Ray-casting Engine [Menon and Robinson, 1993].

Abdel-Malek and Yeh presented a formulation to solve the general case of geometric entities of multiple parameters [Abdel-Malek and Yeh, 1997]. The formation takes into consideration parametric limits of the surfaces imposed in terms of inequality constraints. It is difficult to implement this method in cases where there are a higher number of entities because as the number of non-linear equations resulting from the method increases, the equations become more difficult to solve.

Blackmore and Leu [1990, 1992] introduced a new approach, called the sweep differential equation (SDE) method, for the study of swept volumes. This approach fully exploits the Lie group structure of the set of Euclidean motions, thereby enabling the problems of involving swept volumes to be reformulated in the context of differential equations. The potential of this approach for automated manufacturing applications has been discussed and demonstrated in robot motion planning [Blackmore et al., 1992b; Deng et al., 1994, 1996].

The SDE theory was implemented initially for two dimensional objects. A computer program was developed by Jiang [1993] to model the swept volumes of polygonal objects undergoing general planar sweeps. A set of candidate points was

calculated by the SDE method and trimmed to form the boundary of the swept volume. Qin et al. [1994] modified Jiang's work by introducing an envelope differential equation method along with the sweep differential equation method to generate the grazing points for the swept-volume boundary more efficiently. Wang et al. [1995, 1996] used the SEDE (Sweep-envelop differential equation) method to represent the swept volumes generated by a 7-parameter APT tool under general motion in NC machining and implemented it in 5-axis NC machining simulation and verification.

Blackmore and Leu also extended the SDE method to the modeling of deformed swept volumes and described its implementation [Blackmore et al., 1994; Leu et al., 1998].

The two most significant applications (NC simulation and verification and robot motion planning) that have been done using the SDE method for deformed and undeformed swept volumes are off-line applications. So the computation time was not a particularly critical issue in the implementation of the SDE. All the implementation strategies that have been developed for the SDE method attempted to create a stand alone program for computing the swept volumes and relied on existing CAD/CAM packages (Pro/Engineer) to perform the simulation, Boolean operation and verification. As a result, the potential of the SDE for playing a role in accelerating the processes of simulation, verification and especially Boolean subtraction had not really been adequately explored. An important contribution of our research is the development of a new algorithm for Boolean subtraction in a VR environment that demonstrates how the SDE method can be used to perform on-line operations in real time. And, of course, as mentioned above, this real-time capability is essential for the present method. This new algorithm for Boolean

subtraction is specifically adapted to the use of the Sweep Differential Equation (SDE) method. It integrates the SDE method with ray-casting and graphics techniques to perform real-time Boolean subtractions for objects in the VR environment.

4.4.2 Sweep Differential Equation

The object M under consideration is assumed to be closed and bounded with a boundary surface ∂M which is piecewise smooth. In practical terms, the object considered has a smooth boundary except for a finite number of edges and vertices.

A rigid sweep is a continuous (actually smooth in most cases) family of rigid motions. Mathematically, a smooth sweep σ in which R is the field of real numbers is a smooth mapping:

$$\sigma : [0,1] \rightarrow E(n) \quad (4.1)$$

where $E(n)$ is the Lie group of Euclidean motions such that σ at $t=0$, denoted by σ_0 , is the identity mapping. From this definition, if we let σ_t represent a sweep at time t , it can be written as

$$\sigma_t(x_0) = x(t) = \xi(t) + B(t)x_0 \quad (4.2)$$

where $\xi : [0,1] \rightarrow R^n$ and $B : [0,1] \rightarrow SO(n)$ are smooth functions representing the translation and rotation of the sweep, respectively; x_0 represents any point on the boundary of object M in R^3 .

The set

$$\sigma_t(M) = M(t) = \{ \sigma_t(x); x \in M \} \quad (4.3)$$

is called the t -section of M under the sweep σ . The swept volume of M generated by σ is

$$S_\sigma(M) = \{ \sigma_t(M) : 0 \leq t \leq 1 \} \quad (4.4)$$

Solving equation (4.2) for x_o and substituting it into the time derivative of Equation (4.2), we can obtain the sweep vector field (SVF)

$$X_\sigma = \dot{x}(t) = \dot{\xi}(t) + \dot{B}(t)B^T(t)(x - \xi(t)) \quad (4.5)$$

We apply the boundary flow method (BFM) [Wang, 1997] for the representation and calculation of swept volumes by partitioning the boundary of ∂M at each t into ingress, egress and grazing points. The set of ingress (egress) points of $M(t)$, denoted by $\partial_- M(t)$ ($\partial_+ M(t)$), consists of all points $x \in \partial M(t)$ at which $X_\sigma(x, t)$ points into (out of) the interior of M . Those points that are neither ingress nor egress points are called grazing points and denoted by $\partial_0 M(t)$, as shown in Figure 4.3

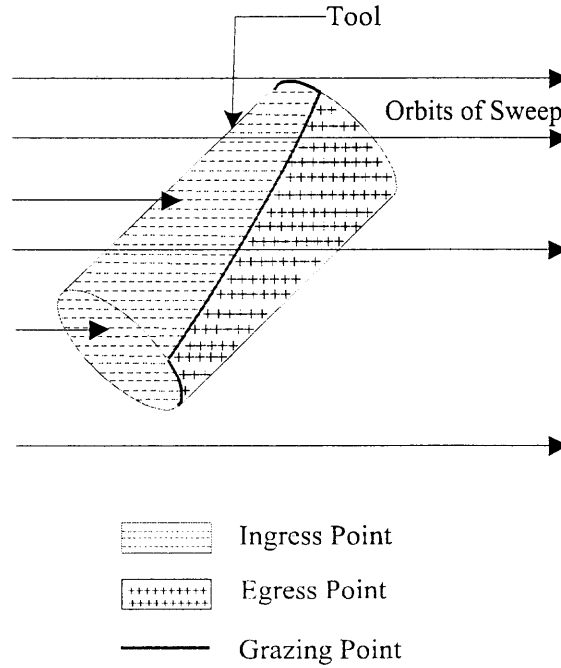


Figure 4.3 Partitioning the boundary of an object

Mathematically, The above notions can be defined in the context of the tangency function. The tangency function for a sweep of object M is defined as

$$T(x, t) = \langle X_\sigma(x, t), N(x, t) \rangle \quad (4.6)$$

where $\langle a, b \rangle$ denotes the inner product of a, b in R^n , and $N(x, t)$ is the unit outward normal vector on the smooth part of M at the point of concern. For the points on a smooth surface, we have:

$$\begin{aligned}\partial_- M(t) &= \cup \{T(x, t) < 0, x \in M, t \in [0, 1]\} \\ \partial_+ M(t) &= \cup \{T(x, t) > 0, x \in M, t \in [0, 1]\} \\ \partial_0 M(t) &= \cup \{T(x, t) = 0, x \in M, t \in [0, 1]\}\end{aligned}\tag{4.7}$$

Let the object M and sweep σ be as above; the boundary of the swept volume is given by $G(M) \setminus W(M)$, where $G(M) = \partial_- M(0) \cup \partial_+ M(1) \cup \{\partial_0 M(t) : 0 < t < 1\}$ is the candidate boundary set which consists of the ingress points of object M at $t=0$, egress points of M at $t=1$, and all the grazing points between $t=0$ and $t=1$. $W(M)$ denotes the trimming set which belongs to the interior of some t -section of M , and thus it does not belong to the portion of the swept-volume boundary. For the points not on a smooth surface, ingress, egress and grazing points can also be defined easily by using the tangency function.

A fairly straightforward method for calculating the outward normal vector field can be used. Let a surface in three-dimensional space be defined parametrically in terms of the equation (in terms of the real parameters u and v):

$$x(u, v) = \begin{bmatrix} x(u, v) \\ y(u, v) \\ z(u, v) \end{bmatrix}$$

If $x(u, v)$ is the vector of any point on this surface, the outward normal vector of the surface at this point is

$$N(x, t) = \frac{\partial x(u, v)}{\partial u} \times \frac{\partial x(u, v)}{\partial v}\tag{4.8}$$

where \times denotes the standard vector cross product. The tangency function, therefore, can be obtained from

$$\begin{aligned} T(x, t) &= \langle X_\sigma(x, t), N(x, t) \rangle \\ &= X_x(y_u z_v - y_v z_u) + X_y(x_v z_u - x_u z_v) + X_z(x_u y_v - x_v y_u) \end{aligned} \quad (4.9)$$

where x_u, x_v, y_u, y_v, z_u and z_v stand for

$$x_u = \frac{\partial x(u, v)}{\partial u}, \quad x_v = \frac{\partial x(u, v)}{\partial v} \quad \dots$$

and X_x, X_y , and X_z represent the x, y and z components of $X_\sigma(x, t)$.

4.4.3 Development of Computer Program

In our implementation of the VR environment for creation of freeform surface, the virtual tools used to remove material in the design model creation process are limited to NC milling cutters in order to insure that the created design model can be obtained by real NC machining. The most advanced NC machine motions are limited by 5-degrees-of-freedom (5-DOF) because of the symmetric geometry of the tools and the inherent limitations of the machine configurations. So we will limit the motion of the virtual tool to 5-DOF to simulate the NC machines. Although different machines have different motion types, generally we assume roll and pitch motions of the tool with respect to a general coordinate system. As we can see in Figure 4.4, the tool axis vector z , whose orientation is defined, can be transformed by rotation of angle α about the x axis and rotation of angle β about the y axis.

A computer program in C++ for the generation of virtual tool swept volume for flat-end milling has been developed. The CL data, which indicates tip cutter location of the virtual tool, are obtained from the tracker motion device. These data are used to generate the tool motion equations by linearly interpolating the movements of the individual cutter axes because CL data of the tracker motion is fed to the VR environment in a step format. The CL data representing tool tip position and tool orientation are expressed by $(x_c, y_c, z_c, i_c, j_c, k_c)$ where (x_c, y_c, z_c) stands for the cutter tip position and (i_c, j_c, k_c) stands for the cutter orientation. The position of a point on the cutter during machining can be expressed as:

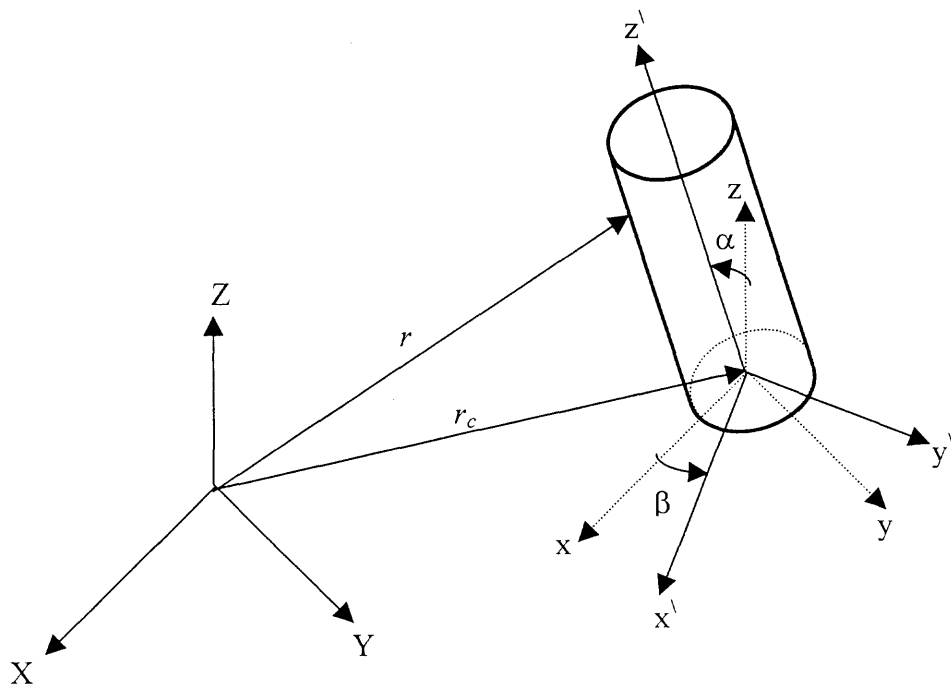


Figure 4.4 Coordinate frame transformation

$$r = \begin{pmatrix} x(t) \\ y(t) \\ z(t) \end{pmatrix} = \begin{pmatrix} x_c(t) \\ y_c(t) \\ z_c(t) \end{pmatrix} + B \cdot \begin{pmatrix} x_0 \\ y_0 \\ z_0 \end{pmatrix} \quad (4.10)$$

where (x_0, y_0, z_0) is a vector representing the initial tool tip position, and B is the rotational transform matrix depending on the roll (β) and pitch (α) angles as follows:

$$B = \begin{pmatrix} \cos \beta & -\sin \alpha \cdot \sin \beta & \cos \alpha \cdot \sin \beta \\ 0 & \cos \alpha & -\sin \alpha \\ -\sin \beta & \sin \alpha \cdot \cos \beta & \cos \alpha \cdot \cos \beta \end{pmatrix} \quad (4.11)$$

where

$$\begin{aligned} \alpha &= -\tan^{-1}(j_c(t) / \sqrt{i_c^2(t) + k_c^2(t)}) \\ \beta &= \tan^{-1}(i_c(t) / k_c(t)) \end{aligned} \quad (4.12)$$

Thus, given one block of CL data:

$$(x_c(0), y_c(0), z_c(0), i_c(0), j_c(0), k_c(0)) \rightarrow (x_c(1), y_c(1), z_c(1), i_c(1), j_c(1), k_c(1))$$

The CL data $(x_{c(t)}, y_{c(t)}, z_{c(t)}, i_{c(t)}, j_{c(t)}, k_{c(t)})$ between the initial and final locations can be calculated using linear interpolation. Substituting the obtained $(i_{c(t)}, j_{c(t)}, k_{c(t)})$ into equation (4.12) and then equation (4.11), we obtain the rotation transform matrix. Together with the translation vector $(x_{c(t)}, y_{c(t)}, z_{c(t)})$, the sweep differential equation is completely defined for the block of cutter motion.

Using the described approach (SDE) for cutter swept volume representation, a software program for the generation of the swept volumes, called SWEEPVR, was developed. The basic process of computing swept volumes is by the procedure described in the following program:

- 1 **Input of data by the user:** Tool geometry information, which are the length (L) and the radius (R) of the tool.

2 **Read data from tracker motion:** cutter location (CL) $(x_c, y_c, z_c, i_c, j_c, k_c)$ where (x_c, y_c, z_c) represents the virtual tool tip position and (i_c, j_c, k_c) stands for the tool orientation.

3 **From time =0 to time = 1 do** the following:

3.1. **Compute the interval CL** data from the initial tool position and final position.

Since data from the tracker motion device are discrete linear interpolation is used as follows:

$$x_c(t) = x_c(0) + (x_c(1) - x_c(0))t \quad (4.13.a)$$

$$y_c(t) = y_c(0) + (y_c(1) - y_c(0))t \quad (4.13.b)$$

$$z_c(t) = z_c(0) + (z_c(1) - z_c(0))t \quad (4.13.c)$$

$$i_c(t) = i_c(0) + (i_c(1) - i_c(0))t \quad (4.13.d)$$

$$j_c(t) = j_c(0) + (j_c(1) - j_c(0))t \quad (4.13.e)$$

$$k_c(t) = k_c(0) + (k_c(1) - k_c(0))t \quad (4.13.f)$$

where t is the time.

3.2. Compute the derivative of equations (4.13) with respect to the time, thus we have:

$$\dot{x}_c(t) = x_c(1) - x_c(0) \quad (4.14.a)$$

$$\dot{y}_c(t) = y_c(1) - y_c(0) \quad (4.14.b)$$

$$\dot{z}_c(t) = z_c(1) - z_c(0) \quad (4.14.c)$$

$$\dot{i}_c(t) = i_c(1) - i_c(0) \quad (4.14.d)$$

$$\dot{j}_c(t) = j_c(1) - j_c(0) \quad (4.14.e)$$

$$\dot{k}_c(t) = k_c(1) - k_c(0) \quad (4.14.f)$$

3.3. Compute the rotational transform matrix (B) from equations (4.12) and (4.11).

3.4. Compute the derivative of the transform matrix with respect to time.

$$B = \begin{pmatrix} -\sin \beta \cdot \dot{\beta} & -\cos \alpha \cdot \sin \beta \cdot \dot{\alpha} - \sin \alpha \cdot \cos \beta \cdot \dot{\beta} & \cos \alpha \cdot \cos \beta \cdot \dot{\beta} - \sin \alpha \cdot \sin \beta \cdot \dot{\alpha} \\ 0 & -\sin \alpha \cdot \dot{\alpha} & -\cos \alpha \cdot \dot{\alpha} \\ -\cos \beta \cdot \dot{\beta} & \cos \beta \cdot \cos \alpha \cdot \dot{\alpha} - \sin \beta \cdot \sin \alpha \cdot \dot{\beta} & -\sin \alpha \cdot \cos \beta \cdot \dot{\alpha} - \cos \alpha \cdot \sin \beta \cdot \dot{\beta} \end{pmatrix} \quad (4.15)$$

where $\dot{\beta}(t)$ and $\dot{\alpha}(t)$ represent the time derivatives of the roll (β) and pitch (α) angles, respectively.

3.5. Discretizing

Triangulate the boundary of the virtual tool linearly with triangles of diameter $< m^{-1}$ for some integer $m \gg 1$. The sweep vector field will be computed for each vertex of the triangle. The scheme for triangulating the virtual tool, which has been used in this program is as follows:

- 1- The virtual tool has been sliced along the z-axis of the coordinate system of the virtual tool. In other words, the length of the virtual tool has been divided as shown in Figure 4.5.
- 2- Each slice of the virtual tool has been discretized on the lateral boundary of

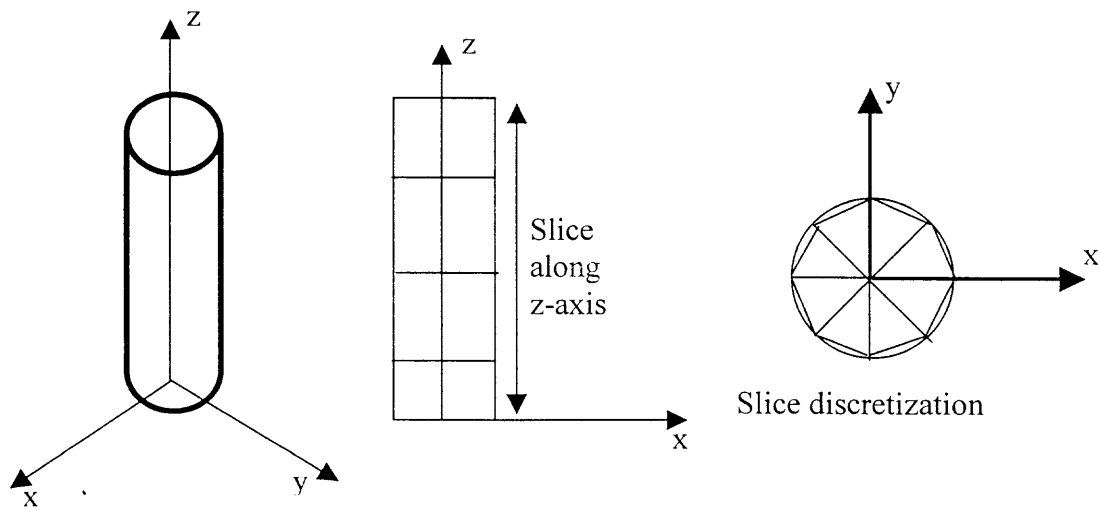


Figure 4.5 Virtual tool discretization

the cylinder.

The computing process has been done as follows:

- From slice $t=0$ to slice $t = N$ and at time $= t$ do the following:
 1. Compute the tangency function at angle $= 0$.
 2. Check the value of tangency function at time $=t$, slice $= n$ and angle $=0$.
Use the boundary flow formula in equation (4.7) to determine if the first point is an ingress or egress or grazing point.
 3. From angle $=10^\circ$ To 360° with respect to the positive x-axis and measured counter-clockwise do as follows:
 - Compute tangency function at each angle.
 - Check the tangency function value according to the boundary flow formula to determine if the point is egress, ingress or grazing.
 - If the point at (angle-1) is ingress and the point at (another angle) is an egress point or the point at (angle-1) is egress and point at (another angle) is an ingress point then there is a grazing point between the (angle-1) and (another angle). In this case subdivision is performed until the tangency function value is within the tolerance for a zero value of the tangency function.
 - Once a particular point on the lateral boundary of a slice is found to be a grazing point the point is recorded by using the output function.

3.6 Tangency Function value computation:

- 1- Compute the position of vertex points of the triangulation in the local coordinate system of tool by using the simple equation (4.16) :

$$\text{Vertex } (x) = (\text{radius of the tool}) \cos (\text{angle}) \quad (4.16.a)$$

$$\text{Vertex } (y) = (\text{radius of the tool}) \sin (\text{angle}) \quad (4.16.b)$$

$$\text{Vertex } (z) = (\text{length of the tool}) / (\text{slice number}) \quad (4.16.c)$$

- 2- Compute the outer normal vector in the local coordinate system of the tool by using the following equations (4.17):

$$\text{Normal } (x) = (\text{radius of the tool}) \cos (\text{angle}) \quad (4.17.a)$$

$$\text{Normal } (y) = (\text{radius of the tool}) \sin (\text{angle}) \quad (4.17.b)$$

$$\text{Normal } (z) = 0 \quad (4.17.c)$$

- 3- Compute the global outer normal by multiplying the local normal in equation (4.16) by the transform matrix in equation (4.11).
- 4- Compute the vector field by using the SDE function.
- 5- Compute the dot product between the vector field obtained from step 4 and global outer normal vector from step 3.
- 6- Return the value of the tangency function of the result in step 5.

3.7 Compute the SDE function for vector field:

- 1- Compute the derivative of the transform matrix by equation (4.15).
- 2- Multiply the result of step 1 by the position vector of the tool from the tangency function.
- 3- Compute the derivative of the position vector of the tool by equation (4.14).
- 4- Add result step 3 and step 4 to get the vector field for that point.
- 5- Return the result of step 4 to the tangency function.

4. **Output function for the ingress, grazing, egress points** in the triangulation representation is as follows:

1. Transfer points information from the local coordinate system to global coordinate by using the equation (4.15).
2. Add the position vector obtained by using equation (4.13) to the result of step 1.
3. Results of step 2 are the global coordinate positions of the grazing points.
4. The method that has been used for connecting curves of the grazing points is as follows:
 - Create a half space plane containing the x and z-axes between the two center points of two successive slices.
 - If the line that connects two grazing points on successive slices does not intersect the half space plane, then it is an edge of an admissible triangle. If the line does intersect the half space plane, a new admissible connecting line is chosen as illustrated in Figure 4.6.

From the above description of the algorithm that we implemented, it is found that the SDE method computes the swept volume of a moving object in real time with good accuracy. It provides a boundary representation for the swept volume. In this method, the sweep differential equation representing the velocity field is obtained by taking the time derivative of the motion equation which includes both translation and rotation.

The boundary of an object is partitioned (at any time) into ingress, egress, or grazing points. To determine whether a point is an ingress, egress, or grazing point, one can simply compute and interrogate the inner product(s) of the velocity vector and the outward normal vector(s) at that point. We have shown that the grazing points on the boundary of a three-dimensional object form a curve, which is one-dimensional, and that the boundary of the object's swept volume is formed essentially by joining the grazing curves along the sweep. Thus, the SDE method reduces the complexity of finding swept volumes by two dimensions, therefore it is very efficient.

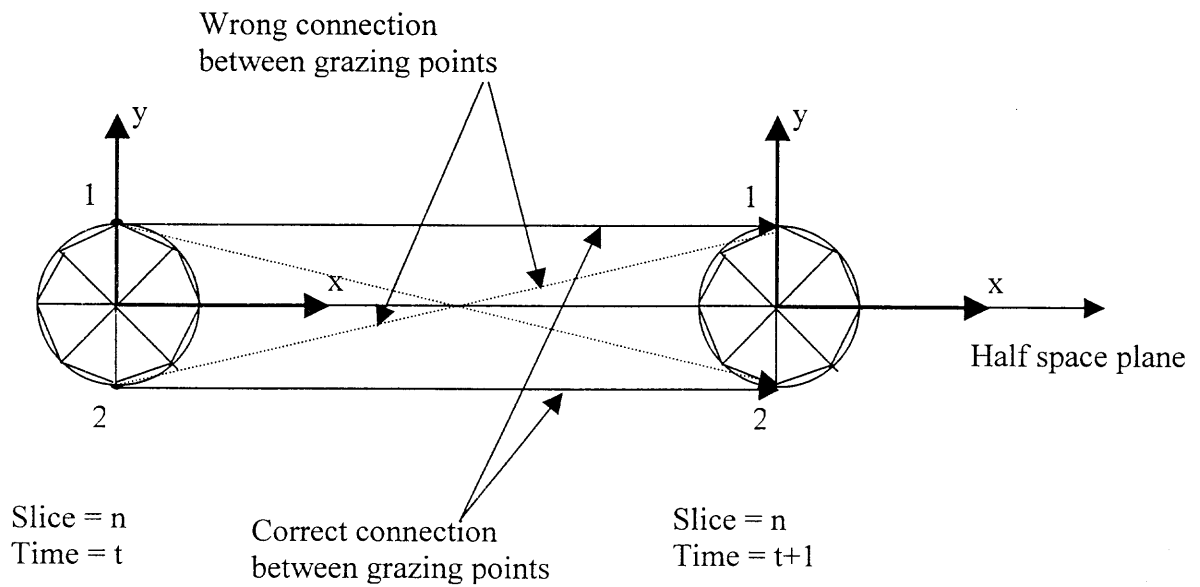


Figure 4.6 The connection of grazing points

4.5 Solid Modeling Using the Ray-Casting Technique

4.5.1 Related Studies

The ray-casting method is a spatial partitioning representation approach. It has been used as the basis for modeling, constructing, and displaying solids. The idea of ray-casting apparently originated with Goldstein and Nagel [1971]. In this paper, ray-casting is called ray tracing and it is used to obtain shaded pictures of geometric objects. The ray-casting technique has also been used to create pixel images of objects [Leu et al., 1986]. All of the approaches [Atherton et al., 1987; Van Hook, 1986; Wang and Wang, 1986] use a projection of the workpiece shape in a fixed viewing direction and along with variation of z-buffer for visualizing the machining process. Huang and Oliver [1994] used a dixel representation derived from the dixel structure of Van Hook. A similar data structure is used by Sito and Takahashi [1991] in their G-buffer method.

Many experts in the CAD/CAM field have doubts about the sufficiency of ray-casting and consider it to be an impractical, brute force method. Most of the applications of the ray-casting method have been limited to off-line verification of NC and robot programs.

To maintain the advantages and overcome limitations of previous work on ray-casting, this research adopts the spatial partitioning approach as the basis for a comprehensive system which also incorporates the advantages of the discrete vector intersection. The goal is to develop a platform independent of the virtual creation system that is capable of providing satisfactory freeform surfaces and objects with good accuracy in real-time.

In the interest of completeness, the fundamentals of the ray-casting method are described in the sequel. We then describe some innovations aimed at extending the range of the application of ray-casting and accelerating its implementation.

4.5.2 Basis of Ray-Casting

The concept of ray-casting comes from the camera model and the following text defines the camera model, describes the information ray-casting provides, and explains how the information is applied.

Light rays and camera geometry form the basis for all geometric reasoning here. The pinhole camera model is used, as this is the standard in image processing [Roth, 1978]. This camera model consists of a focal point (or eye point) and a square pixel array (or screen). The visible solids, called the "scene" are in the front of the camera-enclosed

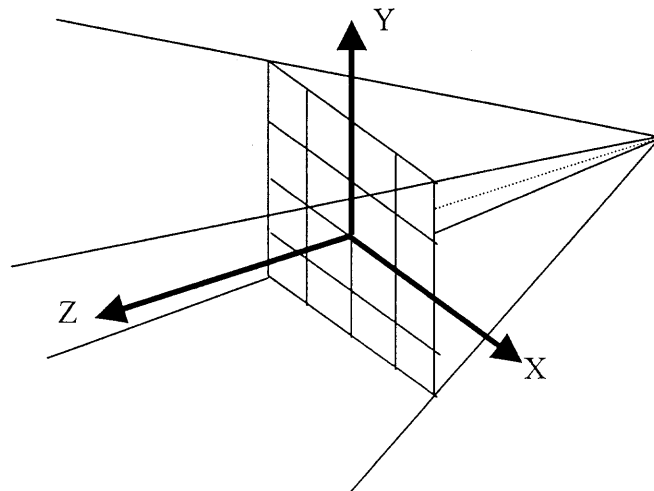


Figure 4.7 Camera model and local coordinate system

by the truncated, four-sided pyramid shown in Figure 4.7. Straight light rays pass through the pixel array to contact the focal point with the scene, where it is assumed that there is

one ray per pixel. To shade pictures, the ray's intensities are measured and stored as pixels. The reflecting surface determines the value of the pixel with its corresponding ray intersections with each pixel's ray. For convenience, the standard coordinate system for the camera has the screen in the X - Y plane, the scene in the $+Z$ half space, and the focal point on the $-Z$ axis.

A ray is simply a straight line in the 3-D space of the camera model. It is most conveniently defined in parameterized form in terms of a point (X_0, Y_0, Z_0) and a direction vector (D_x, D_y, D_z) . In this form, points on the line are ordered and accessed via a single parameter t . For every value of t , a corresponding point (X, Y, Z) on the line is defined:

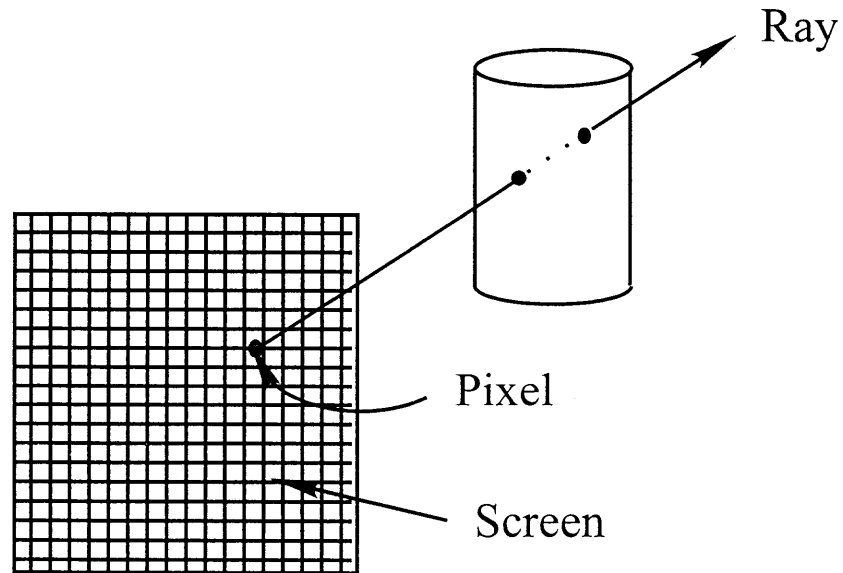


Figure 4.8 Example of ray-casting

$$\begin{aligned} X &= X_0 + D_x t \\ Y &= Y_0 + D_y t \\ Z &= Z_0 + D_z t \end{aligned} \tag{4.18}$$

so, a ray in a parallel view that passes through pixel (X, Y) in the screen is simply defined as $(X, Y, 0)$ given the screen center $(0, 0, 0)$ and eye point $(0, 0, +Z)$.

A computer program in C++ for implementing the ray-casting method called RAYCASTING has been developed. RAYCASTING is a ray-solid evaluator that is the heart of the solid modeling system. Input to RAYCASTING is a ray; output from RAYCASTING is information about how the ray intersects the scene. Knowing the camera model and the solid in the scene, RAYCASTING finds where the given ray enters and exits the solid (Figure 4.8).

This information is returned in a list as shown in Figure 4.9. Where I and J are integers indicating the position of the ray on the screen. The order list of ray parameters Z_i , denote the enter-exit points. The ray enters the solid at point Z_1 , exits at Z_2 , enters at Z_3 , etc., and finally exits at Z_n . Here n is the number of the ray-solid intersections. The point Z_1 is closest to and Z_n is furthest from the camera. For the associated ray parameters, RAYCASTING also returns a list of pointers to the surfaces through which the ray passes.

Based on this information from RAYCASTING, several applications, including the computation of volumes, can be carried out. The volume of a solid bounded by curved surfaces is easily computed by the "approximating sums" integration method, by approximating the solid with a set of rectangular parallelepipeds. This is accomplished by taking an "in-depth" picture of the solid in a parallel view. Casting the rays through the screen into the solid partitions the solid into volume elements. Two dimensions of the parallelepipeds are constant, defined by the 2-D spacing of rays in the screen. The third dimension is variable, defined by enter-exit points returned by RAYCASTING.

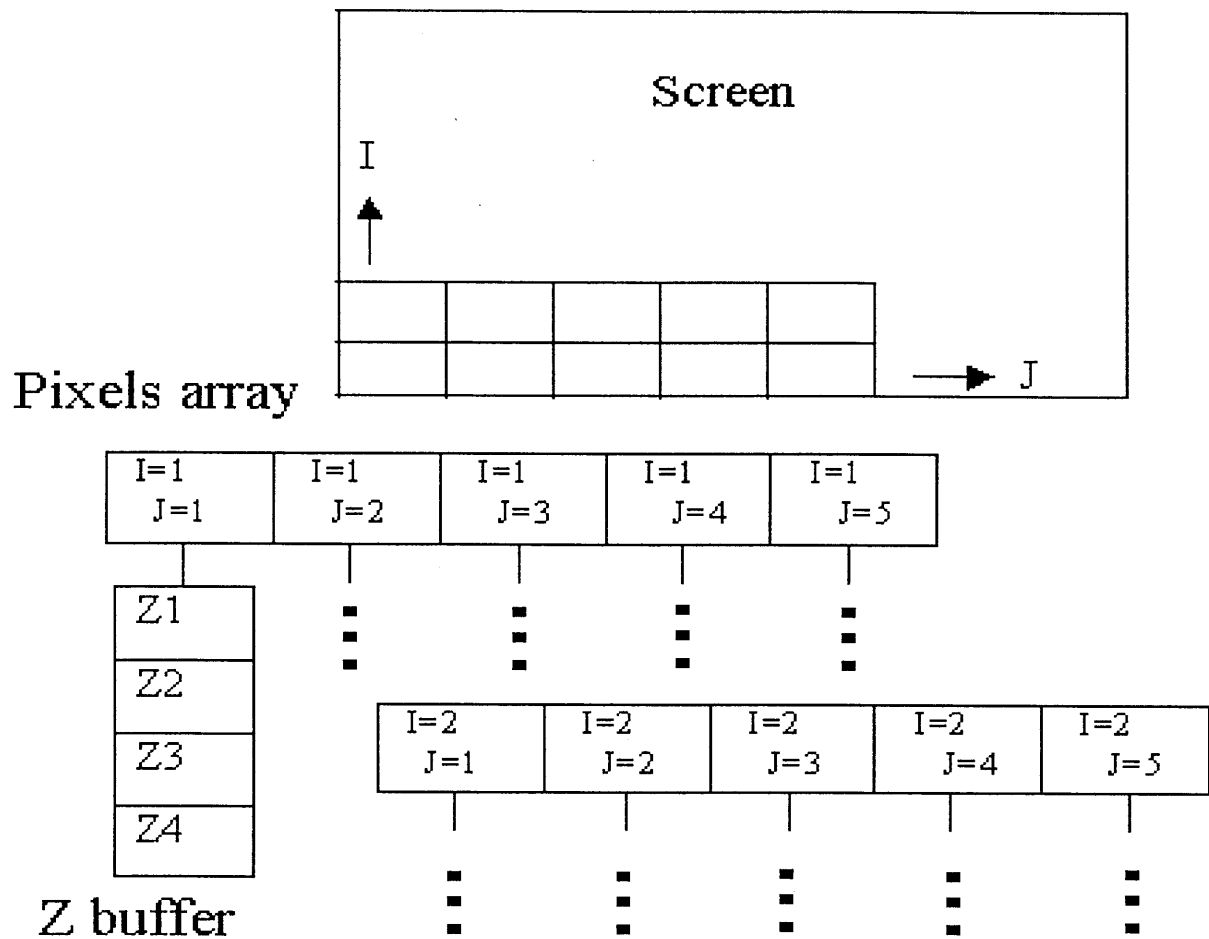


Figure 4.9 Data structure of ray-casting

4.5.3 Ray-Casting Algorithm

RAYCASTING, the ray-solid evaluator, finds where a given ray enters and exits a given solid, accounting for the combined operators Union (U), Difference (D) and Intersection (I). Before presenting the mechanics of RAYCASTING, the data structures for solid compositions will be described. Then we describe the coordinate systems that

RAYCASTING uses and explain how rays are transformed from one coordinate system to another. Finally the RAYCASTING algorithm is presented in detail.

The data structure representing solid compositions is an inverted, simple binary tree as shown in Figure (4.1). Leaf nodes of the tree are primitives and the internal nodes are composites; solids are formed by combining the operators Union (U), Difference (D) and Intersection (I). If a composition has N primitive solids, then there are $N-1$ composite solids for a total of $2N-1$ solids. This relation holds regardless of whether the tree is balanced or not. All solids in the tree, both composites and primitives have a number of attributes. For example, the Boolean operations are identified by a simple sequence of integers. There is a local to scene transform attribute, which is a 4×4 linear transformation matrix, and a scene to local transform attribute which transforms points from the local coordinate system of the primitive to the scene coordinate system

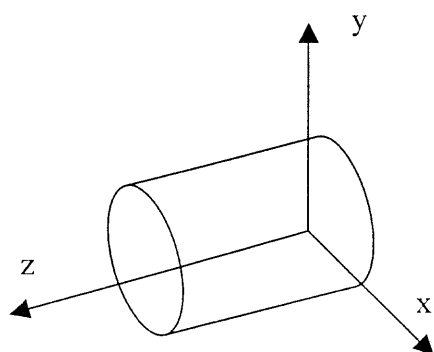
RAYCASTING uses three kinds of coordinate system, all Cartesian: the scene (or global) coordinate system; local coordinate systems of the primitives; and the screen coordinate system. The user should be aware of only the global coordinate system. The user can move the objects in the global coordinate system and the other coordinate systems are used in order to simplify geometric calculations with the camera model or with the primitives.

Each primitive has a local coordinate system. When a user creates a new primitive solid, its position and scale in the global coordinate system are the same as those in the local coordinate system. When the user moves or scales the primitive, however, its position or size changes in the global coordinate system. The scene-to-local transform

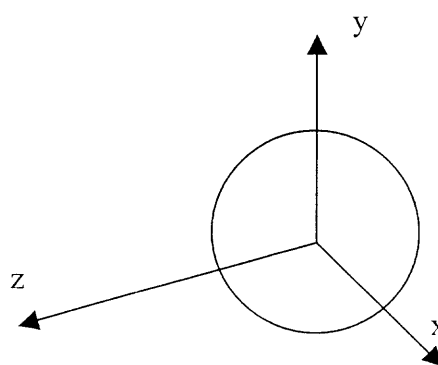
records the transformation needed to convert it back to the local coordinate system. The local coordinate systems of the primitives are shown in Figure 4.10.

In the screen coordinate system, the screen is in the X - Y plane. That is, the equation of the screen is $Z=0$. The scene is in the $+Z$ half space and eye point is in the $-Z$ half space, displayed from the center of the screen. The coordinate systems of the screen and scene are linked via the "screen-to-scene" transform and its inverse "scene-to-screen" transform, as defined by the camera model. Each primitive has its scene-to-local transform and inverse, but there is only one screen-to screen transform and inverse. Given a ray originating in the screen coordinate system, RAYCASTING transforms it into the local coordinate systems of the primitives via the scene coordinate system in order to find the ray-solid intersection points.

Using the homogeneous coordinate techniques common to computer graphics and geometric modeling, linear transformations between the coordinate systems are readily defined [Foley, 1997]. Only point and vector transformations are important for ray



(a) The cylinder with unit radius and length.



(b) The sphere with a unit radius

Figure 4.10 Local coordinate system of primitives: (a) The cylinder's axis lies along the Z -axis with one end at origin, (b) The sphere is centered at the origin

casting. The equation used for a point transformation is:

$$P^1 = P M \quad (4.19)$$

Where $P = (X, Y, Z)$ is the given point, M is a 4×4 linear transformation matrix and $P^1 = (X^1, Y^1, Z^1)$ is the transformed point that results. While the ray, which is a line, is transformed by simply transforming its fixed point and direction vector:

$$\begin{aligned} (X_0, Y_0, Z_0) (D_x, D_y, D_z) M &= ((X_0, Y_0, Z_0, 1) M) ((D_x, D_y, D_z, 0) M) \\ &= (X_0^1, Y_0^1, Z_0^1) (D_x^1, D_y^1, D_z^1) \end{aligned} \quad (4.20)$$

where (D_x, D_y, D_z) is the vector defining the line and (D_x^1, D_y^1, D_z^1) is this vector after transformation. When transformed this way, line parameterizations are the same in both coordinate systems.

Swept volumes are converted to triangulation representation as described previously. These triangles are transformed by using the scene to local transforms. The effects of scaling, rotating and translating a solid are achieved by scaling, rotating and translating the rays.

Given a ray and a solid composition tree, RAYCASTING classifies the ray with respect to the solid and returns the classification to the caller. By definition, the classification of a ray with respect to a solid is the information describing the ray-solid intersection. It designates what parts of the ray are in the solid versus out of the solid.

In what follows, we describe in greater detail the two essential parts of the algorithm: (1) Intersecting rays with swept volumes and (2) Combining the results of Boolean operations.

The general ray-solid intersection problem reduces to ray-triangles intersection problems because rays enter and exit solids via the solid surfaces and all surfaces are

triangulated. The result of intersections of the ray with swept volumes has four possible outcomes:

1. The ray missing the swept volume.
2. The ray tangent to the swept volume at a single point.
3. The ray enters and exits the swept volume at some points.
4. The ray lies on a face of the swept volume.

In cases 1 and 2, the ray is classified as being completely out and no intersection points are noted. In case 3 and 4, two or more intersection points are noted and the ray is divided into three segments: out-in-out.

The ray-triangle intersection is performed by considering the plane in which the triangle lies. The intersection of the parameterized ray $P(t) = P(0) + Dt$ with the plane is given in point-normal form as $(X - Q) \cdot V = 0$, where V is a normal vector to the plane, Q is a given point on the plane and X is any point on the plane. Then we simply replace the X by $P(t)$ and solve :

$$(P(0) + Dt - Q) \cdot V = 0 \quad (4.21)$$

Solving this equation for t gives:

$$t = \frac{(Q - P(0)) \cdot V}{D \cdot V} \quad (4.22)$$

and the intersection point is at

$$P(0) + \frac{Q - P(0)}{D \cdot V} \cdot D \quad (4.23)$$

After finding the intersection point of the ray and plane, it is necessary to determine whether the point is within the boundary of the triangle or not. To solve this problem we have developed the following half space method (Figure 4.11):

1. Number the three vertices of the triangle by P_1 , P_2 and P_3 and the intersection point by P_0 .
2. Create two vectors; V_{01} between the intersection points P_0 and P_1 ; V_{12} between the points P_1 and P_2 respectively.
3. Find a unit normal vector N_{012} resulting from the cross product of the two vectors in step 2.
4. Repeat steps 2 and 3 for the rest of the points of the triangle to find normal vectors N_{023} and N_{031} .
5. If the direction of the three normal vectors is the same then the intersection point lies within the triangle, otherwise the point is in the exterior of the triangle.

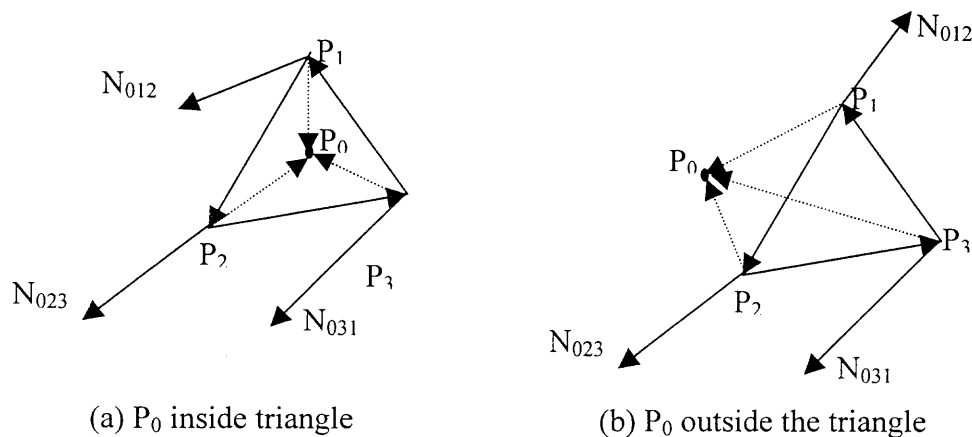


Figure 4.11 Check the intersection point for inclusion in the triangle:
 (a) all normal vectors in the same direction (b) normal vector " N_{012} " is not in the same direction as normal vectors " N_{023} " and " N_{031} "

4.5.4 Boolean Operations by Ray-Casting Technique

We will use the ray casting method for Boolean operations because it has the following advantages: 1) The data size is linearly proportional to the number of pixels, and 2) The Boolean operation is one-dimensional. The advantage of the data size has been implemented through our data structure as defined in previous subsections. The other advantage of the Boolean operation is described in this section.

Figure (4.12) illustrates how to combine ray classification from near (N) to far

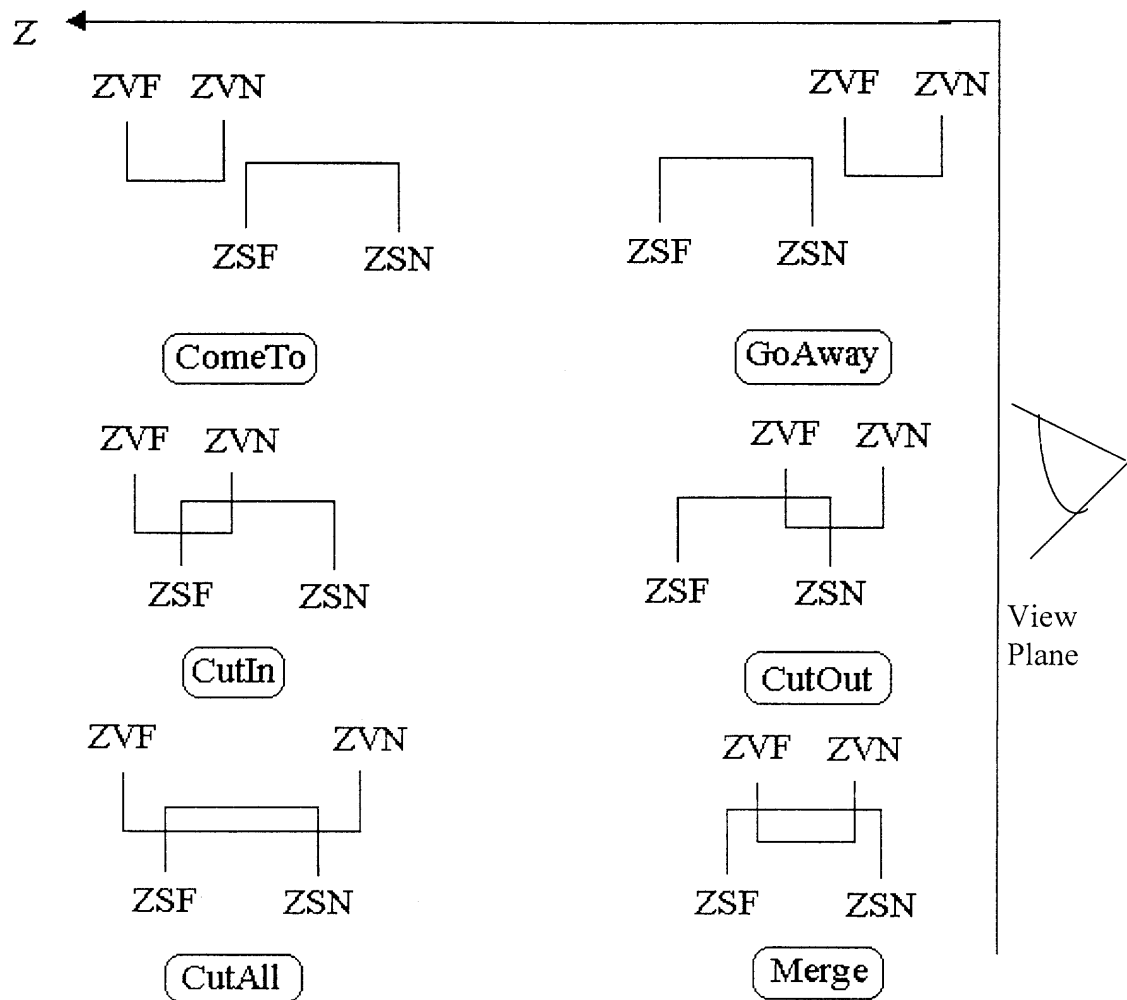


Figure 4.12 Classification of Boolean operation cases

(F). It is clear we have six cases for the Boolean operations. For the first two cases: "ComeTo" means the virtual tool is moving toward the workpiece for cutting but yet has not intersected the workpiece; "GoAway" means that the virtual tool is moving away from the stock. In both of these cases, we do not need to change the data structure of the stock. The ZSF stands for the Z value of the Stock when it is far from the view plane and the ZSN stands for the Z value of the Stock when it is near the view plane. The ZVF stands for the Z value of the swept Volume when it is far from the view plane and the ZVN stands for the Z value of the swept Volume when it is near the view plane. For the "ComeTo" and "GoAway" cases, the data structure does not change because there is no intersection between the stock and the swept volume.

In the "CutIn" case, the data structure of the stock is updated by replacing the value of the ZSF by the value of the ZVN. But in the case of "CutOut", the value of the ZSN is replaced by the value of the ZVF. In the case of "CutAll", the values ZSF and ZSN are both replaced by null. The last case, which is "Merge" is different from the other cases and is obtained by adding the points ZVF and the ZVN to the data structure of the stock. After adding the points to the data structure, then the new data structure for the link list will look like ZSN-ZVN-ZVF-ZSF.

4.5.5 The Cost of Ray-Casting

Ray-casting qualifies as a brute force method for solving problems. The minimal ray casting algorithm is very simple, particularly in consideration of its many applications and ease of use. But most applications call ray-casting in their inner loop, an inner loop that is executed thousands or hundreds of thousands of times.

By introducing additional logic and refinements to the minimal ray-casting algorithm, it will run much faster and solve the real time problem. In this section we first explain that the complexity of the scene, measured by the number of swept volumes in the solid composition, is related to ray-casting CPU usage. Then we explain how the use of three graphics and data mechanism techniques will speed up the ray-casting computations.

In the minimal ray-casting algorithm presented in section 4.4.2, memory and CPU usage is directly proportional to the scene complexity-that is, to the number of swept volumes or primitive solids in the composition. Memory usage is relatively small, so it is not a concern. The important question is "How fast is it?".

To appreciate the computation cost of using ray-casting, consider the following scenario. A user has built a model of a complex solid using 300 swept volumes and asks the Boolean operation program to display the solid on a raster display. Assume that the Boolean operations program uses ray-casting in a simple, straightforward way. Assume the raster display has 600×600 pixels, the Boolean program generates 360 000 rays and passes them one at time to the ray casting program for in-out classification. The cost of visiting each solid is $300 \times 360\,000 = 108\,000\,000$ recursive procedure calls. Each ray-solid intersection test involves one or more ray- surface intersection tests. Assuming the swept volumes have an average of 12 triangles, every call to ray-casting involves $300 \times (12-1)$ ray-surface intersection tests: $300 \times (12-1) \times 360\,000 = 1188\,000\,000$ ray -surface intersection tests. When two ray classifications are combined, the two ray-solid intersection lists are merged in sorted order. The computation is simple, but it is performed at every composite solid, so the worst case is N^2 for a composition with N

solids. The following discussion assumes the combined cost is linear, like the other costs. At each composite the solid, ray-casting combines the ray classifications from far and near with cost: $300 \times 360\,000 = 108\,000\,000$ classification combinations. Last of all, for each solid, ray-casting transforms the ray into local coordinate systems of the primitives for ray-solid intersection test that cost: $300 \times 360\,000 = 108\,000\,000$ ray transformation.

So the total cost for generating the Boolean subtraction of a solids is the sum of the costs:

- 108 000 000 recursive procedure calls.
- 1188 000 000 ray -surface intersection tests.
- 108 000 000 classification combinations.
- 108 000 000 ray transformation.

Although this might be a good place to wait for tomorrow's faster computers or to seek special hardware for this simple, highly repetitive task, there are easy ways to significantly reduce the cost. The next section shows how the cost can be dramatically reduced for almost all compositions.

4.5.6 Reduction of the Cost and Acceleration

The cost of ray-casting will be reduced by using three techniques which are the improved bounding box technique, the scan-rendering technique, and a new type of data transformation mechanism.

Improved Bounding Box Technique

Detection of ray intersection for each unit is a critical step in realizing fast Boolean operations. The traditional bounding box is simply an orthogonal rectangular parallelepipeds in the screen coordinate system (Figure 4.13). It is represented in the

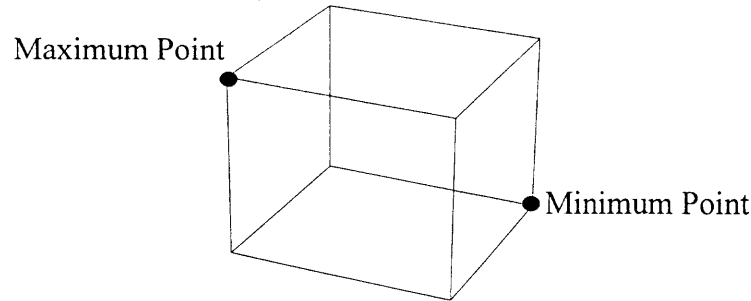


Figure 4.13 Traditional Bound box

screen coordinate system because rays originate there and they are constrained by the camera model to pass through the screen. A box is defined by six numbers, the minimum and maximum X , Y and Z 's of the solid. This information is stored in each solid. The ray intersection test is basically a point in the rectangle test. So for a given ray passing through pixel (X, Y) in the screen, the test is simply 2-D because rays usually start at the screen and extend to infinity in the scene. When the rays are bounded in depth, as they are in our application, the conditions applied are:

$$\begin{aligned} X_{min} &\leq X \leq X_{max} \\ Y_{min} &\leq Y \leq Y_{max} \\ Z_{min} &\leq Z \leq Z_{max} \end{aligned} \tag{4.24}$$

The steps to compute a bound box around a primitive in the screen coordinate system are:

1. List the vertices of the polyherdron that tightly encloses the primitive in its local coordinate system.

2. Transform the polyhedron's vertices from the local coordinate system to the screen coordinate system.
3. Project the transformed vertices to the screen. Since the view is parallel, then (X, Y, Z) simply projects to $(X, Y, 0)$. This step can be deleted if the screen coordinate system is the global coordinate system.
4. Find the minimum and maximum values of the X, Y coordinates of the projection and of the unprojected Z coordinate. These define the bounding box.

By employing the projection of the swept volume and its bounding box, we find that the bounding boxes reduce the number of the computations of the ray-casting method by the ratio: (number of pixels in bounding box)/ (screen pixels). The value of this ratio varies from zero, when there is no intersection between the solids and one if the size of bounding box is equal to the size of the screen. This ratio is on average less than 0.10. Consequently the reduction of cost will be considerable if bounding boxes are used.

This bounding box can be further improved by making it fit more tightly to the swept volumes. The shape of the swept volumes are generated by the VR environment as in Figure 4.14.

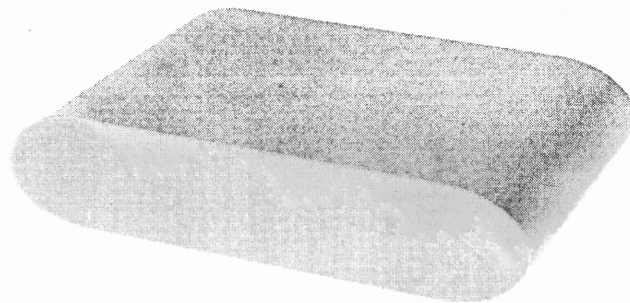


Figure 4.14 The swept volume shape

The projection of the swept volume onto a plane is shown in Figure 4.15. The traditional bounding box that covers the swept volume will be quite a bit larger than the projection as shown in Figure 4.15.

The improved bounding box adapts the traditional bounding box to the natural geometry corresponding to the SDE method for computing and representing the swept volumes. In particular the bounding boxes are fitted to the regions where SDE computations are performed and then moved with the swept volume. The main steps for

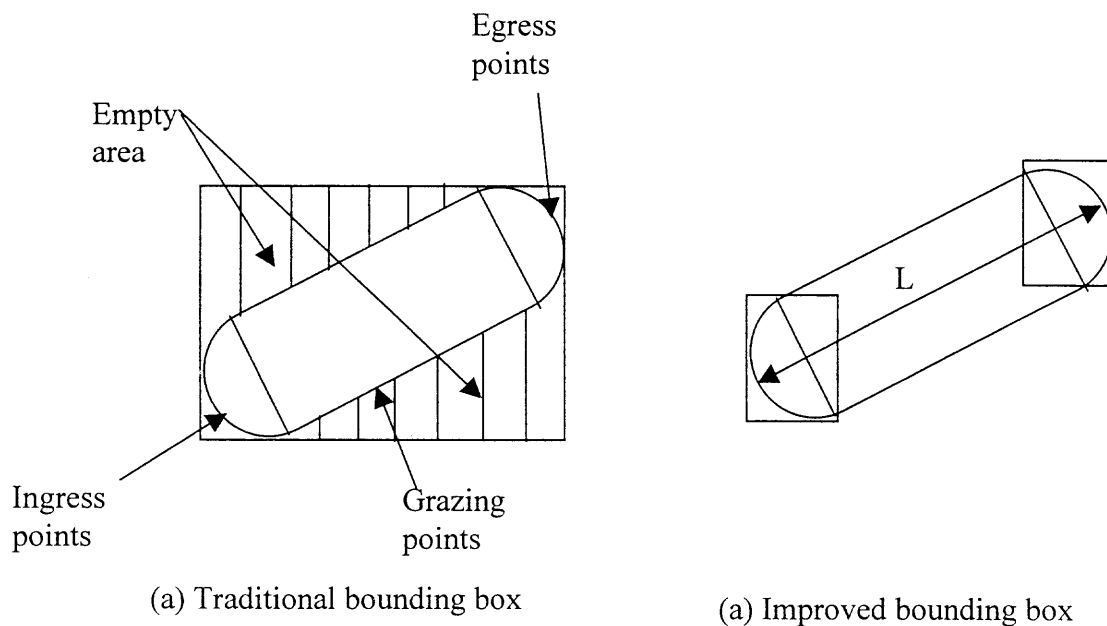


Figure 4.15 The improved and traditional bounding boxes

improving bounding boxes are as follows:

1. List ingress points that represent the initial points of the swept volume in the local coordinate system.
2. Transform the ingress points from the local coordinate system to the screen coordinate system.

3. Project the transformed ingress points to the screen. Since the view is parallel, then (X, Y, Z) simply projects to $(X, Y, 0)$. This step can be deleted if the screen coordinate system is the global coordinate system.
4. Find the minimum and maximum values of the projection coordinates X , Y and of the unprojected Z s. These define the bounding box.
5. Repeat steps 1 to 4, for each set of points (grazing and egress) generated by the SDE method.

One can readily see that this improved bounding box technique leads to a reduction in the computational cost of ray-casting by a factor of $1/L$, where L is the diameter of the swept volume in the projection plane.

Scan Rendering Technique

Scan-rendering can be regarded as a modified ray-casting method. It differs from the ray-casting method in that the ray entry-exit information in relation to the solid is obtained line-by-line, rather than pixel-by-pixel. As shown in Figure 4.16, a scan plane skims through one row of pixels (one scan line) of the screen to a polyhedral solid in the scene. The intersection points between the scan plane and each edge of the solid can be obtained by solving simultaneously the equations representing the scan plane and the edge. For a polyhedral face which intersects with the scan plane, two intersecting points can be found for the edges surrounding the face, from which two pixels corresponding to the intersecting points and associated depth information (x -values) can be found. The depth information for the in-between pixels in relation to this face can be obtained by linear interpolation. After repeating the same process for each scan plane and for each face, the

entry and exit points for the rays passing through the entire screen can be obtained. In comparison with the ray-casting method, scan rendering generally requires less computation time, but is limited to polyhedral solids, which is what we employ for creating freeform surfaces and objects in the VR environment. In fact, all objects in our VR environment are represented by a triangulated boundary representation. Hence, the element of our representation is a triangle.

The main steps in our scan rendering technique (Figure 4.16) are as follows:

1. List the the polyhedral (triangular) faces that represents the solid in its local coordinate system.
2. Transform the polyhedral faces from the local coordinate system to the screen coordinate system.
3. Project the transformed polyherdral vertices onto the screen. Since the view is parallel, (X, Y, Z) simply projects to $(X, Y, 0)$. This step can be deleted if the screen corrdinate system is the global corrdinate system.

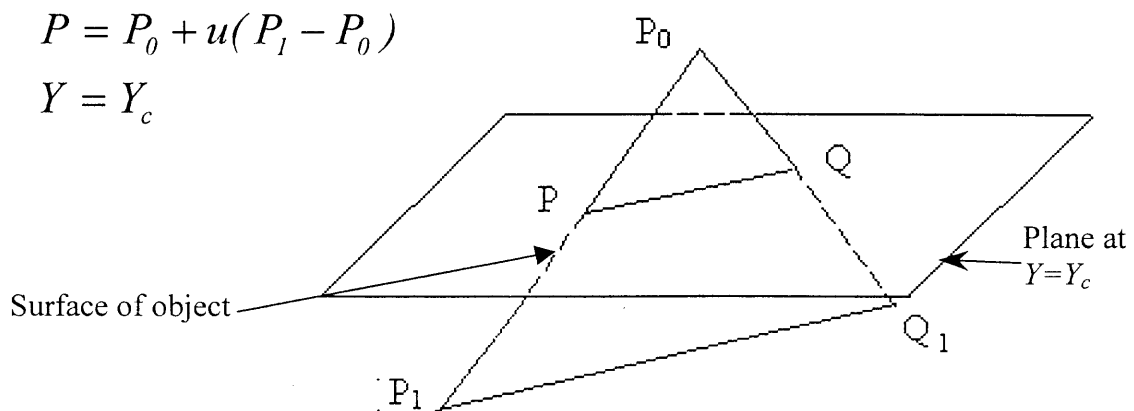


Figure 4.16 The scan rendering technique

5. Find the minimum and maximum values of the projected X and Y coordinate and of the unprojected Z coordinate. These define the minimum and maximum values of Y_c .
6. Selecte the two edges of the triangle that intersect the plane, and represent the edge lines in parametric form with parameter u and solve the equation set: $P(t) = P_0 + u (P_1 - P_0)$ and $Y = Y_c$.
7. Obtain the two intersection points from step 6, then do the linear interpolation between them, after sorting, to find the set of values of pixels in the row Y_c .

This scan rendering technique reduces the number of computations of the ray-casting in terms of ray-surface intersection cost. The ray-surface intersection cost is propotional to the number of the polyhedral faces multiplied by the number of edges. This represents a big improvement over the simple ray-casting technique where ray-surface intersection cost is propotional to the number of the rays in the screen display multiplied by the number of polyhedral faces. This is because the number of pixels is typically at least an order of magnitude larger than the number of faces of the object boundary triangulation.

Data Transformation Mechanism

The last technique to speed up the ray-casting technique consists in changing the mechanism of transformation of the data between different coordinate systems. In the traditional ray-casting technique, intersection points are computed in the primitive coordinates, which requires transformation of the ray representation from the screen to primitive coordinates. These vectors (rays in parametric representation) are transformed from the screen to the world, and then from the world to the primitive coordinate system

by multiplying with 4×4 matrices representing the homogeneous transformations (Figure 4.17). Since the number of rays is huge, the computation of transformation

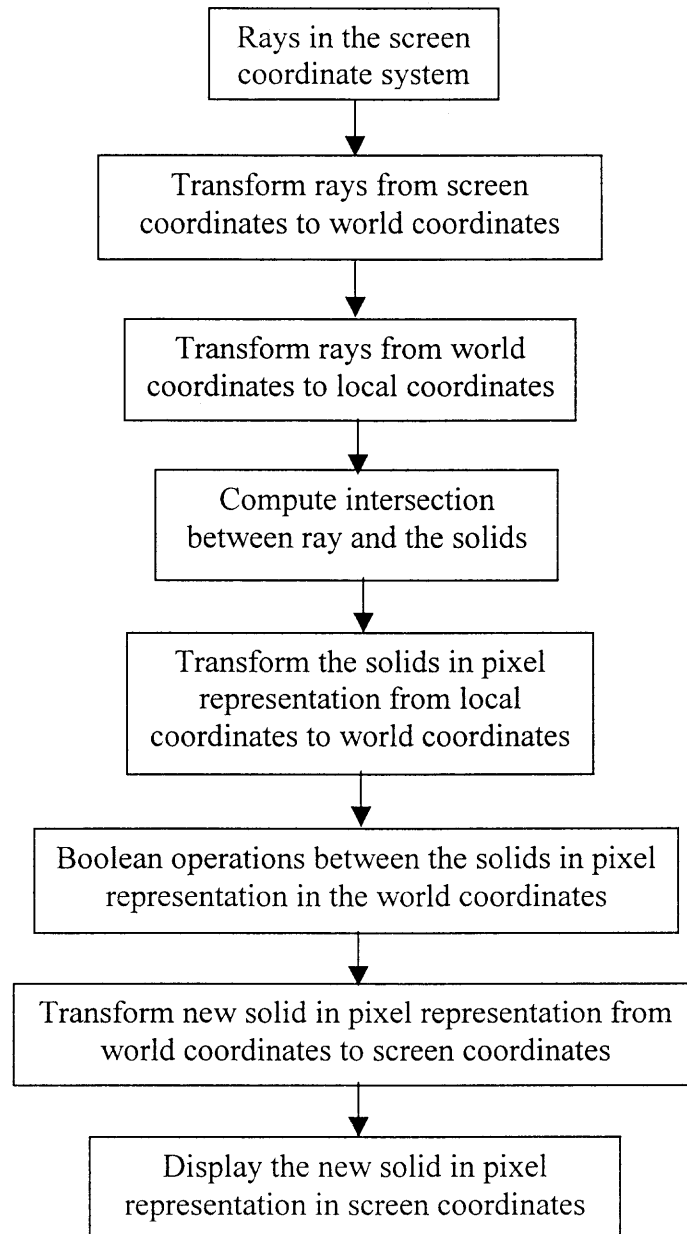


Figure 4.17 The traditional transformation mechanism in ray-casting

processes reduces the ray-casting efficiency.

To decrease the transformation computation and speed up the ray-casting technique, one can reverse the transformation process. Since the number of points that represent the solid is very small compared to the number of vectors (rays), the primitives are transformed from the primitive local coordinate system to the screen coordinate system. After this transformation is complete, ray-surface intersection is applied. The flow chart of the new mechanism is shown in Figure 4.18.

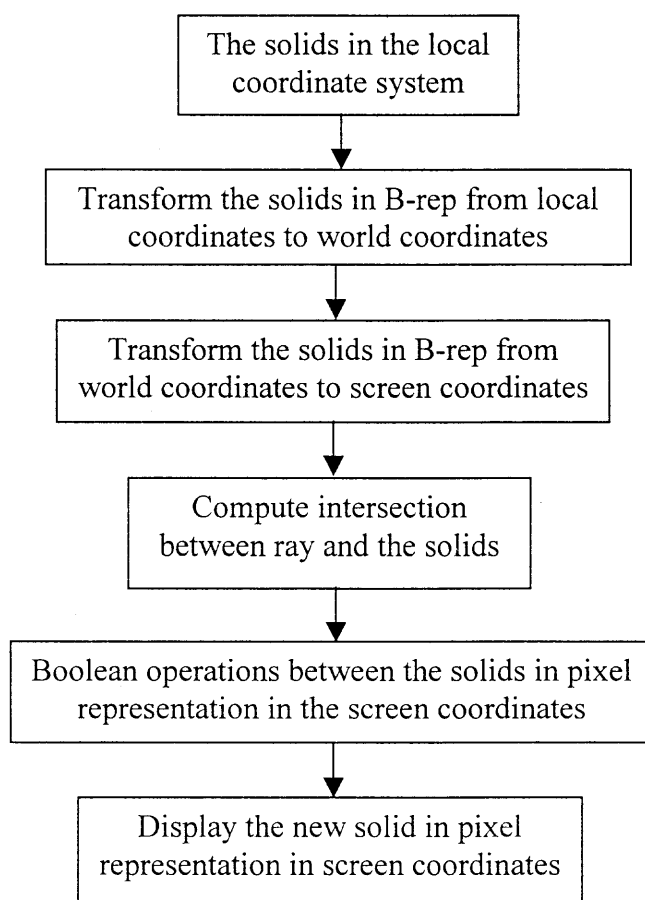


Figure 4.18 New transformation mechanism in ray-casting

1. Reduction of the number of transformation processes between the relevant coordinate systems.
2. Reduction of the size of the data in the transformation process, where the data set associated of the rays is much larger than the data set of primitives.

4.6 Summary

An overview of the existing solid modeling methods has been presented. The advantages and disadvantages of existing methods have been discussed. We conclude from the discussion that current CAD/CAM packages can not be used for real-time applications (required in our VR environment). A new solid modeling engine has been described based on the SDE method for computing and representing swept volumes of a moving object in space. The Spatial Partitioning representation (Ray-casting method) is also developed to represent the virtual objects in the VR environment and perform the Boolean operations in real-time.

The implementation of the SDE method, developed and adapted for use in our VR environment, has been described in detail. The traditional ray-casting methods have been discussed and analyzed in terms of CPU usage. The implementation of traditional ray-casting also has been described in detail. Three techniques have been developed and combined with the ray-casting approach to speed up the computation time to meet the real-time requirements.

CHAPTER 5

REPRESENTATION OF THE CREATED DESIGN MODELS

5.1 Introduction

The desired outputs of the VR environment for the creation of freeform surfaces and objects are a solid model envisaged by the designer or engineer that can be manufactured in the real world, together with associated NC machining tool paths. To this end, the VR environment has been described in detail in terms of the VR hardware and user interface in Chapter 3. The components of the solid modeling engine have been presented in Chapter 4 in detail.

The full integration of the Virtual Reality environment for the creation of freeform surfaces and objects in terms of the VR hardware, the solid engine components and the user interface is presented in this chapter. After that, illustrative examples are presented demonstrate how the VR system actually works. The integration of the VR environment with commercial CAD/CAM packages is described. The accuracy of representation of the design model and computation requirements are analyzed and presented in this chapter.

5.2 Integration of the VR Hardware with the Solid Modeling Software

One of the most important parts of this work is to develop a prototype system for the VR environment for the creation of freeform surfaces and objects. This VR system has two main requirements. The first requirement is the flexibility to create a complex object directly and naturally with a computer. This has been achieved through the VR environment. The second requirement is to do the solid modeling operation in real-time without decreasing the flexibility of the VR system. This has been done by developing a platform independent solid modeling engine that is capable of providing realistic Boolean operations, sweeping objects and communicating with the VR hardware in real-time.

In this section, first we describe the integration of the solid modeling components in detail and refer to Chapter 4 in terms of details for each of the components. Then integration of the VR hardware and solid engine software will be presented.

5.2.1 Integration of the SDE Method with Ray-Casting Technique

The solid modeling engine of the VR environment has been built by combining the SDE method and the ray-casting technique. The implementation of SDE has been optimized to determine whether it is needed to generate the egress points or not. Also the ray-casting technique has been accelerated by developing a new bounding box method (improved bounding box method), scan-rendering approach and data transformation mechanism. There are two main issues in the integration of the SDE method with ray-casting as a solid modeling engine. First there is the compatibility of the two methods to communication between each other. The second issue is the data flow between the two

methods so they can achieve the desired real-time computation. This section discusses these issues and how they have been resolved.

The issue of the compatibility means that both methods can input or output a common data structure so that they can exchange data without needing a geometry translator between them. The reason for avoiding the use of a geometry translator between the SDE method and ray-casting is that we can circumvent the following two disadvantages:

1. In general all geometry translators have limitations in the conversion of geometric data from one type to another. This problem occurs in most of the existing CAD/CAM packages.
2. The geometry translator requires considerable computation time to do the translations. Whereas in our VR environment, the computation is greatly reduced.

Avoiding using the geometry translator in the core of our solid modeling engine does not mean that our VR environment can not input and output geometry data from existing CAD/CAM packages. Actually our solid modeling process imports the workpiece geometry information from existing CAD/CAM packages and exports the design model geometry information to existing CAD/CAM packages; this will be described later in this chapter.

It is clear that we need to develop a data structure for use in our VR environments. The data structure should have these properties:

1. The ability to represent the solids in boundary representation because the SDE method generates a swept volume using boundary representation.

2. Easy to use in calculation because the ray-casting technique is used to perform many computations on the geometric data.
3. Simple geometry information for displaying and manipulating.

Our implementation of the SDE method, which has been described in chapter 4, uses triangles to represent the boundary of the swept volumes. Also the intersection calculations of the ray-casting technique on triangles are simple; they just require intersection calculations between lines and planes. In addition, displaying and rendering the triangles is performed very rapidly. The existing accelerator hardware (accelerator graphics cards) can only accelerate the computation for displaying and rendering for triangulation representation. The reason for this is clear because the method used for displaying is based on ray-tracing. As we mentioned previously, ray-casting is based on ray-tracing techniques. In addition to all this, the triangle has a particularly simple geometry that is uniquely determined by three (vertex) points.

The data structure of the triangle that results from the SDE method is represented by a single link list as shown in Figure 5.1. It is known that the link list has several advantageous features including its ability to change sizes dynamically and its sorting efficiency. In the link list shown in Figure 5.1, the triangle (1) is at the head of the link

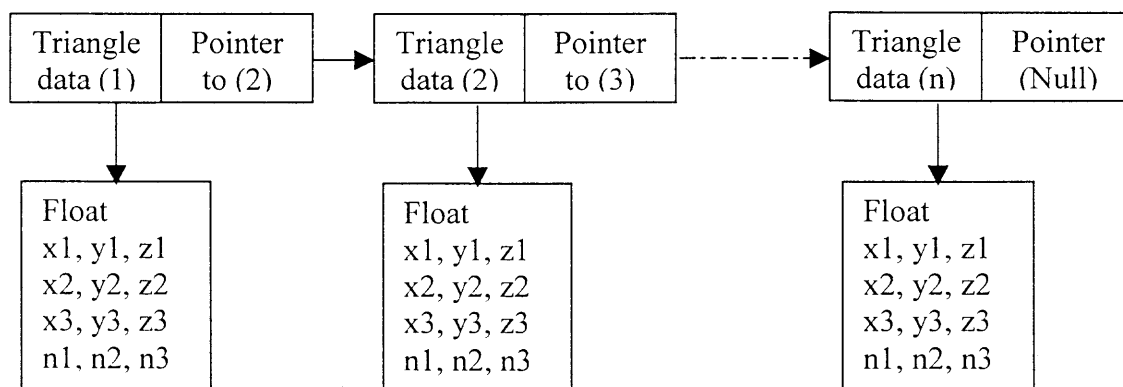


Figure 5.1 The data structure of the swept volume

list and pointing to the next triangle, which is triangle (2). The tail of the link list is the triangle (n) and it points to NULL. Each triangle's data structure includes the vertices of the triangle and a normal vector to each triangular plane. If x_i , y_i and z_i are the position data of the vertex (1) in the triangle, and n_i is a normal vector of that vertex. The normal vector of the vertex in a triangle coincides with a normal vector of the virtual tool during machining and is used to compute the tangency function in the SDE method.

The link list of triangles is used as a queue between the SDE algorithm and ray-casting implementation. A queue is an ordered homogeneous group of items in which the items are added at one end (the back) and are removed from the other end (the front). Queues in computing are very similar to queues in real life. A queue in real life would be a line up at a fast food counter or a bank teller. The people in a queue are dealt with in the order they arrived. In computing this is called a FIFO (first in-first out) queue. Items can only be removed from the queue in the order in which they are added.

Now we will discuss some operations that have been performed on the queue used in our application:

CreateQueue: This operation creates a new empty queue. This operation must be done in order to make the queue logically accessible. This operation is needed to initialize the link list of triangles.

Enqueue: This operation adds a new item to the rear of the queue. This is implemented by adding more triangles to the tail of the link list.

Dequeue: This operation removes, and then returns the item at the front of the queue.

The queue must not be empty. This is implemented to remove the first triangle, which is the head of the link list of triangles.

EmptyQueue: This operation returns true if the queue is empty and false if it is not. The operation is needed to check if the computation the swept volume has been finished or not.

Figure 5.2 shows the data flow between the SDE method and the ray-casting technique using the queue. The SDE method generates a boundary representation of the swept volumes for the virtual tool as a triangulation which is fed into the queue. At the bottom of the queue, the SDE method calls the "CreateQueue" operation to create a queue and initialize the link list and uses the "Enqueue" operation to add triangles to the tail of the link list. At the top of the queue the ray-casting technique calls the "Dequeue" and

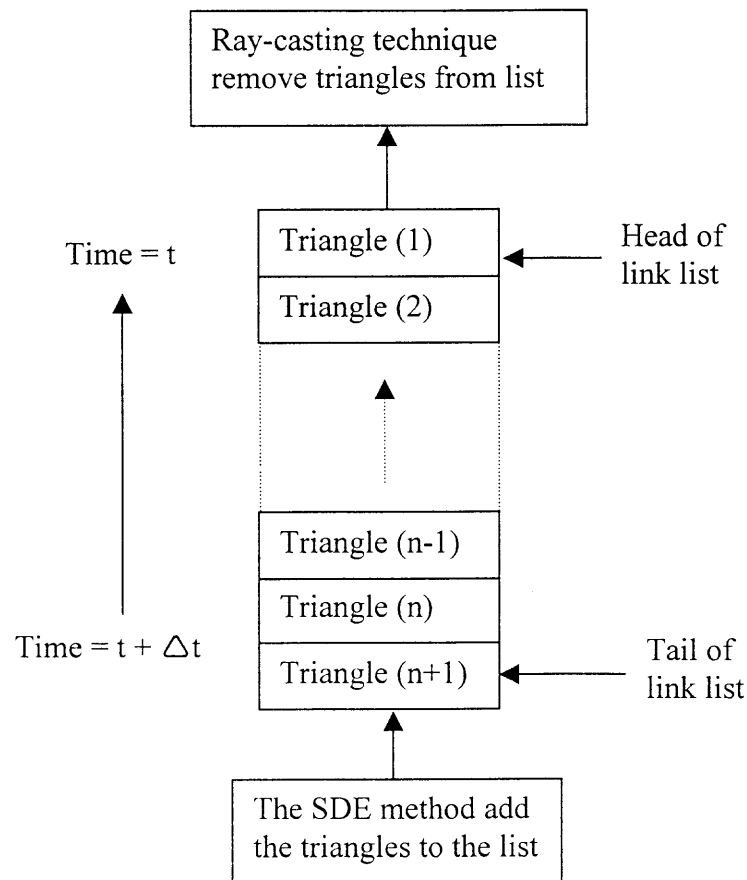


Figure 5.2 The data flow between SDE and ray-casting methods using the queue data structure

"Emptyqueue" operations to remove a triangle from the head of the link list and checks to see if the queue is empty or not. If the link list is empty, then it will be deleted.

Figure 5.3 shows the integration of the components of the solid engine which includes the SDE method, ray-casting, improved bounding box technique and scan-rendering method. The details of these components have been described in Chapter 4. Also the data flow and transformation mechanism is shown in the Figure 5.3. The following paragraphs describe the details of the process and the data flow in the solid modeling engine that achieves real-time computation.

When the VR environment is started, the workpiece is created and represented by a triangular surface patches. The data of the triangles of the workpiece are stored in link list. Then the workpiece (triangles in the link list) is transformed from the world coordinate system to the screen coordinate system. The user places the workpiece in the VR environment (screen coordinate system). The input data from the user is then used to transform the workpiece as the virtual machining is performed.

The improved bounding box technique is applied to the workpiece to determine the pixels that are needed to updated the scene. Since the workpiece is the only object whose topology changes with time, we limit our scene according to the size of the bounding box selected for the workpiece. In other words, the number of pixels for the scene is equal to the number of pixels used to represent the workpiece. Scan-rendering is then applied to each triangle of the workpiece to compute the pixel data information. The data structure of pixels is shown in Figure 4.9. Since the process of the scan-rendering technique is performed on each triangle, each pixel is updated at least twice for a closed swept volume. The requires sorting of the depth values, which are the Z-

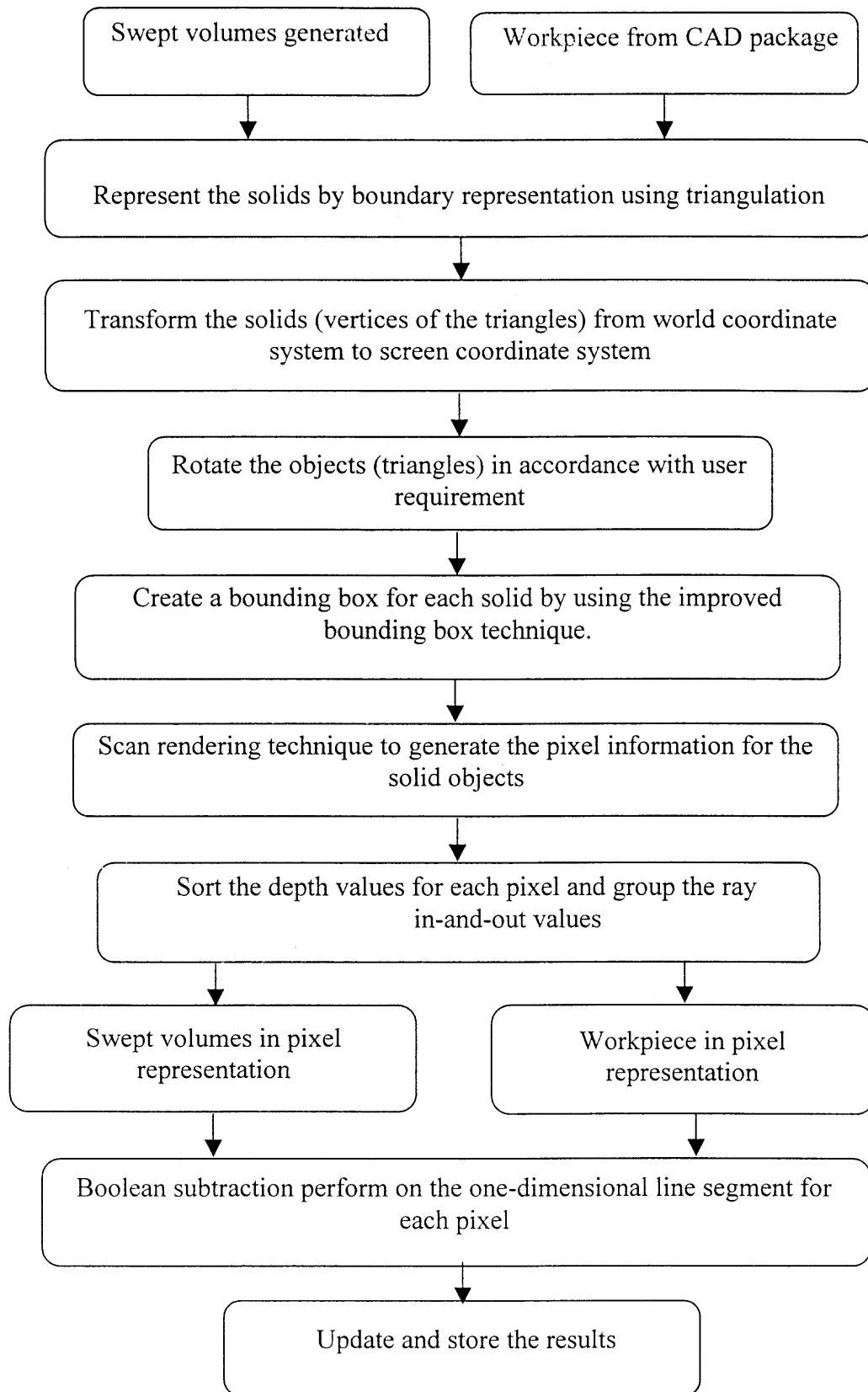


Figure 5.3 Integration of the SDE method with ray-casting technique

values in the screen coordinate system, is done for pixel. After that, the updated workpiece is stored and displayed to the user. The technique that has been developed to display the virtual workpiece and virtual tools will be described later.

It is clear that considerable computation time is required to represent the workpiece. The reason for this is that the size of the workpiece is large in the beginning of the process and requires a large number of pixels to represent it with acceptable accuracy. But we need to keep in mind that this process is the initialization of the VR environment and the user will not notice it because there has not been any interaction with the VR environment.

Swept volumes are generated by the SDE method when the user starts shaping the objects. The triangles of the swept-volume boundary are added to the queue as in Figure 5.2. Because the SDE method generates the triangles of the swept volumes in the world coordinate system, it is necessary to transform these triangles from the world coordinate system to the screen coordinate system (which is the VR environment coordinate system).

The improved bounding box technique is applied to the triangles of the swept volume. To optimize the usage of the improved bounding box technique, we need to develop criteria that determine the minimum size of the geometry to be used for the bounding box. The criteria that have been developed are based on the natural process by which the SDE method generates the swept volume. In the SDE method, the initial ingress points, then the grazing points and finally the terminal egress points are generated. After this sequence of computations, the bounding box is first applied to the ingress points, then the grazing points and lastly the egress points.

The bounding box technique is used for two purposes. First, the bounding box is used to check if the virtual tool intersects with the workpiece or not. This is called collision detection. If the virtual tool (or its swept volume) intersects with the workpiece, the scan-rendering technique is applied. The scan-rendering technique generates the pixel information for the swept volumes of the virtual tool. The pixel information includes the (X, Y) in the screen coordinate system, depth values (Z), and the normal vector of the swept volume surface. Sorting of the Z-values is done for each pixel of the swept volume of the virtual tool. Also the Z-values are classified by in-and-out values.

At this stage, the swept volume of the virtual tool and the workpiece are represented in pixel form and ready for use in the Boolean operation. The Boolean operation is performed in one dimension along a line segment between the workpiece and the swept volume for each pixel. The one-dimensional spaces of application of the Boolean operations are all parallel to the Z-axis of the screen coordinate system. An illustrative example for the Boolean operation is shown in Figure 5.4. It also shows the ray/solid classification.

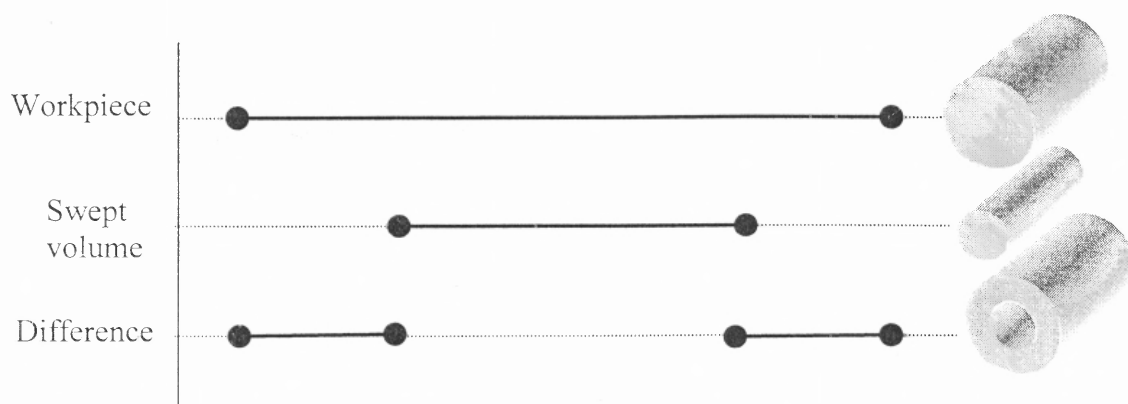


Figure 5.4 One-dimensional Boolean operation

5.2.2 Integration of Solid Modeling Engine with VR Hardware Environment

The VR environment for the creation of freeform surfaces and objects provides the user or designer with several hardware devices to interact with the VR environment. These hardware devices are the 3D-motion tracker device, mouse or keyboard device, monitor device, and audio (speaker) device. The integration of these devices with the solid modeling engine is described in this section.

Figure 5.5 shows the integration of the VR hardware with the solid modeling engine through the design process. The design process begins by activating the VR environment system. The VR system reads the workpiece data from the hard disk and sends it to the solid modeling engine to be converted from boundary representation to pixel representation. While the solid engine is busy performing calculations of the workpiece, the VR system will establish a connection with the motion tracker device and configure it.

Hardware and software configurations are connected to the motion tracker in order for it to work compatibly with the VR system. The hardware configuration is done by using the dipswitches on the ascension electronic unit. Details about the hardware configuration are described in the manual of the Flock of Birds [Ascension Technology Corporation, 1999]. In order for the computer to communicate with the motion tracker, several parameters need to be configured. These parameters are the number of sensors or receivers, number of transmitters, computer port (com1, com2), baud rate (HZ), measurement rate (frame/second) and read and write time for the frame buffer. The first step in this process configuration is the hardware configuration that is completed before running the VR system. It is important to make sure that the parameters have been

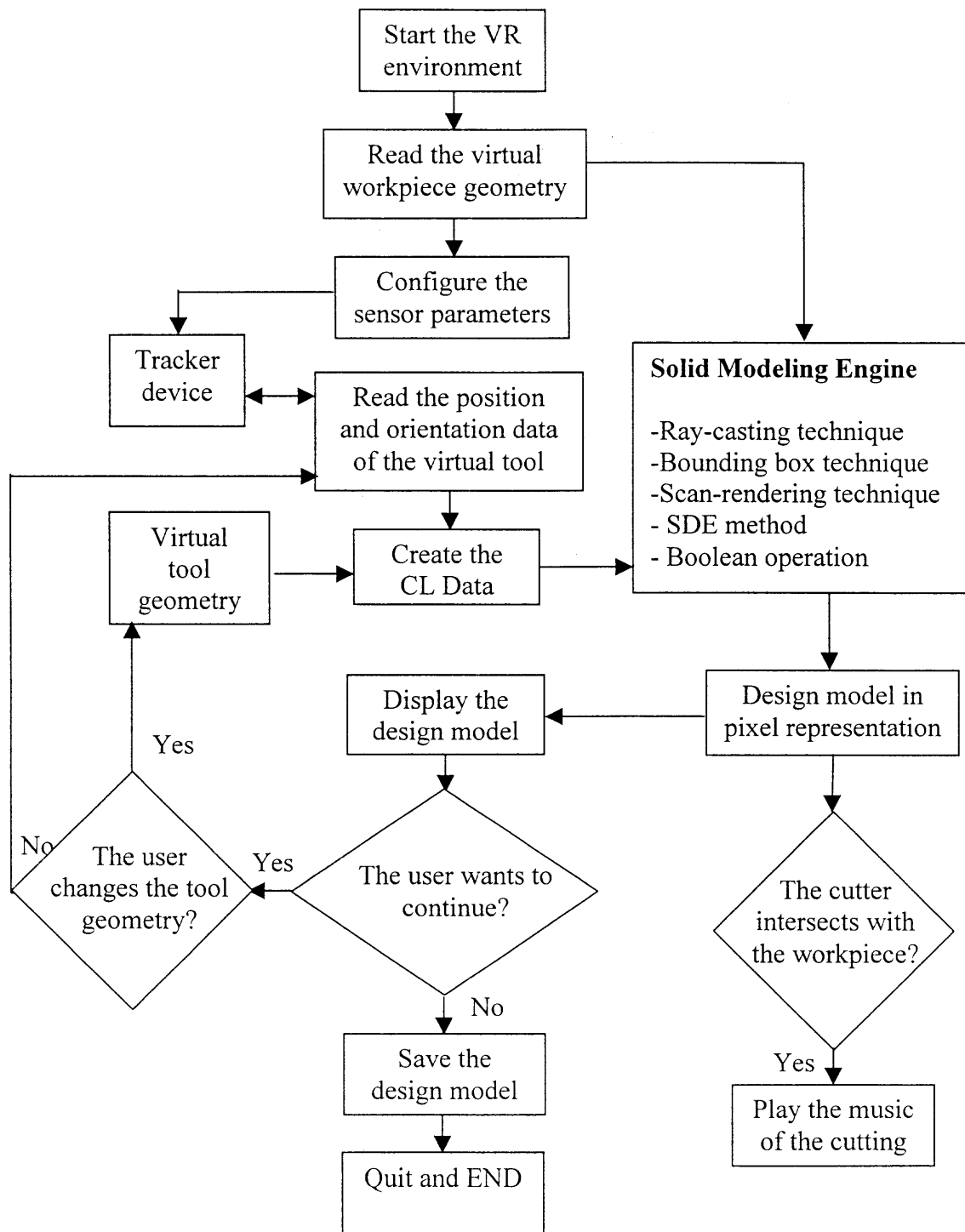


Figure 5.5 The integration of hardware with software

configured correctly in order to establish the right communication between the computer

and the software.

The configuration of the motion tracker device proceeds as follows:

1. Send command to activate the motion tracker device.
2. Display the current system configuration.
3. Send the examine-serial-number command to the ascension electronic unit.
4. Determine the number of devices in the system.
5. Update the system configuration with the new transmitter number and measurement rate.
6. Change the device configuration so that the data format is position/angle.
7. Set up the hemisphere to start reading the data in the upper hemisphere.
8. Start getting continuous or single frames of the position and orientation data from the sensor.
9. Stop getting continuous or single frames of the position and orientation data from the sensor.

If all these steps have been run without any error message then the motion tracker device is ready to communicate with the computer and the software. If any error message has been detected, the manual of the Flock of Birds [Ascension Technology Corporation, 1999] should be referred to debug the program.

After the VR system finishes configuring the motion tracker device, it reads the position and orientation data of the sensor, namely the position and orientation data of the virtual cutter tool. The CL (cutter location) data of the virtual cutter is created from the

position and orientation data of the sensor and the geometry data, comprised of the length and radius of the virtual tool.

The VR system will display the workpiece and the virtual tool to the user. If the user wants to replace the workpiece or change the tool geometry, then the mouse or keyboard can be used to access the virtual reality menu. During the virtual menu display and interaction time, the VR system delays the calculations and operations that are related to the options of the menu. The reason for this is to avoid extra computation time for the workpiece and swept volume and make sure that the virtual workpiece is not intersecting the virtual tool.

When the user is finished manipulating the workpiece and virtual tool, then the virtual carving of the design model can begin. As the user sweeps the virtual tool, the CL data is created and sent to the solid modeling engine. The solid modeling engine will check for collisions between the workpiece and the virtual tool. If there is collision between the workpiece and virtual tool then music will be played by the audio device. Then it will create the swept volumes of the virtual tool from the CL data information by using the SDE method. Boolean operation will be done between the workpiece and swept volumes and the design model is created in pixel representation.

The design model is displayed by using the nearest depth value to the screen. Since the pixel size of the VR system is larger than the pixel size of the screen, we convert the pixel information of the design model to boundary representation. The result of this is a triangulation of the boundary surface of the design model. The technique that we developed for this purpose is as follows:

1. Begin with the pixel with the minimum x and y values $[X(i), Y(j)]$, where i and j are the pixel position in the grid or screen (see Figure 5.6).
2. Obtain two additional pixels: the first one is $[X(i+1), Y(j)]$ which is obtained by incrementing along the x-axis of the screen direction and the second is $[X(i+1), Y(j+1)]$ which is obtained by incrementing along the x-axis and then along the y-axis of the screen.
3. If any point is missing on the grid go to step 6.
4. If j is larger than maximum j , then quit from display.
5. Create a triangle using the three points $([X(i), Y(j)], [X(i+1), Y(j)], [X(i+1), Y(j+1)])$ and display it on the screen.

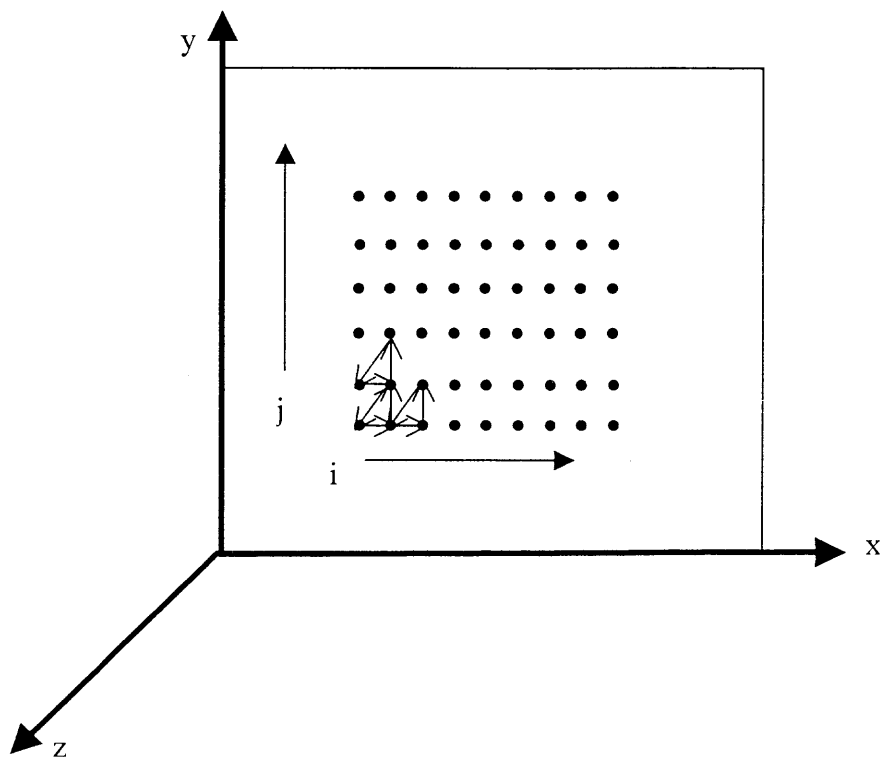


Figure 5.6 Converting from the pixel representation to boundary representation

6. Go to step one and increment i to be $i+1$ if i is not equal the maximum i .

The triangulated boundary surface of the design model is displayed using the OpenGL (Standard graphic Library from Silicon Graphics Corporation SGI) functions and library. In order to display the triangle by using the OpenGL library, we need to provide the position data points and the normal vectors of each point. These already have been computed and exist in the link list. The Graphics Library Utility (GLUT) is used to create the user interface in the form of windows, menus, a mouse and keyboard. For more details about displaying objects using the OpenGL library, the reader can refer to the OpenGL guide book [Neider, et. al, 1993].

If the user continues the designing process and would like to change the tool geometry (length or radius), then this can be done by using the tool menu. In this case, the VR system will automatically do the following: 1) Modify the tool geometry before reading the new data from the motion tracker, 2) Empty the queue for the swept volume and ray-casting technique, and 3) Save the CL data for the old tool geometry. If the user finishes designing the model, the results should be saved before exiting from the VR environment.

It is clear that there are two main parts in our VR system that require most of the overall computation time. The first part is the solid modeling computation time and the second is the display time. The reason for requiring the majority of the computational operations is that they (display and solid engine) are based on the ray-intersection technique (ray-tracing technique). Since the displaying process is separated from the solid engine calculations, the two processes can be performed in parallel. This reduces the

updating time of the VR environment by at least one half of the conventional processing time. Also the user can adjust the speed in cutting material or sweeping the virtual tool according to the feedback that is obtained from displaying the results on the monitor. In other words the user controls the hand speed with which the virtual tool is moved based on the feedback display.

5.3 Integration of the VR System with the Commercial CAD/CAM Packages

The design model created by the our VR system is initially represented by the points on the boundary of the design model that are obtained by computing the intersections of the rays with the boundary of the swept volume of the cutting tool. Next these points must be converted into triangulated surfaces in order to integrate the results of this creation process with existing graphics viewers, like Open Inventor from SGI company, OpenGL, etc.

The triangulated surface is suitable for display but it can not be used as an input to the existing CAD/CAM systems that provide capabilities for computer-aided design, manufacturing, and other engineering applications. For example, many commercial CAD/CAM systems can perform Boolean operations of the created design model with other objects created directly in the CAD/CAM system to form a more complicated model, generate NC machining trajectories for the design model, or perform finite element analysis. Since this conversion of points into triangulated surfaces can not be implemented in these CAD/CAM packages, we need to find some other way to solve this problem.

Pixel or ray representation has been used only for NC simulation and graphics display, but the problem of inputting the pixel information into commercial CAD/CAM systems has received scant attention. Benouamer and Michelucci [1997] presented a method for bridging the gap between the CSG and B-rep by using triple ray representation. They used the ray-casting technique to convert the solid objects from B-rep to Voxel (cubes) by applying the ray-casting method three times from three different independent directions. Then the CSG tree is constructed using these cubes (where the cube is one of the CSG primitives). It is clear that this method works but the computational cost is very high and it is not suitable for our VR system because the computation time would be tripled. Also the size of data file needs to be huge in order to include all the cubes information within the limits of good accuracy.

In this section, we present a method to convert the pixel representation to NURBS (Non-Uniform Rational B-Spline) surfaces, which are well suited for representing the boundary of the created design model.

5.3.1 Overview of NURBS Representation for Curves and Surfaces

There are two fundamental methods for representing curves and surfaces: exact representation by using implicit equations or approximate representation by using polygons or polyhedra. If the curves or surfaces have a simple geometry readily described by implicit equations then exact representation is used. For more complicated curves and surfaces, some form of approximate representation is used.

The complex curves and surfaces are approximated by subdividing them into simple curves and surfaces. The order of the approximation of curves/surfaces can vary from linear approximation by lines/planes to higher order polynomial curves/surfaces.

Cubic polynomials are often used to approximate curves and surfaces.

The cubic polynomials that are used to define a curve in space are of the form

$$\begin{aligned} x(t) &= a_x t^3 + b_x t^2 + c_x t + d_x \\ y(t) &= a_y t^3 + b_y t^2 + c_y t + d_y \\ z(t) &= a_z t^3 + b_z t^2 + c_z t + d_z \end{aligned} \quad (5.1)$$

where the parameter $t \in [0, 1]$, the unit interval. With $T = [t^3, t^2, t, 1]$, and defining the 4×3 matrix of coefficients of the three polynomials as

$$C = \begin{bmatrix} a_x & a_y & a_z \\ b_x & b_y & b_z \\ c_x & c_y & c_z \\ d_x & d_y & d_z \end{bmatrix} \quad (5.2)$$

the equation (5.1) can be rewritten in the following compact form

$$Q(t) = [x(t) \quad y(t) \quad z(t)] = T \cdot C \quad (5.3)$$

We rewrite the coefficient matrix as

$$C = M \cdot G \quad (5.4)$$

where M is a 4×4 matrix basis matrix and G is a four-element column vector of geometric constraints, called the geometry vector. We use the G_x , G_y , and G_z to refer to the x , y and z components of each element of the geometry vector. The geometric constraints are just the conditions, such as endpoints or tangent vectors, that define the curve. Equation (5.3) can be expanded as

$$Q(t) = [x(t) \quad y(t) \quad z(t)] = \begin{bmatrix} t^3 & t^2 & t & 1 \end{bmatrix} \cdot \begin{bmatrix} m_{11} & m_{12} & m_{13} & m_{14} \\ m_{21} & m_{22} & m_{23} & m_{24} \\ m_{31} & m_{32} & m_{33} & m_{34} \\ m_{41} & m_{42} & m_{43} & m_{44} \end{bmatrix} \cdot \begin{bmatrix} G_1 \\ G_2 \\ G_3 \\ G_4 \end{bmatrix} \quad (5.5)$$

The parametric bicubic surfaces are a generalization of the parametric cubic curves. For obtaining $Q(s) = S.M.G$, we allow the points in the geometry vector (G) to vary in 3D along some path that is parameterized on t , so we have

$$Q(s, t) = S \cdot M \cdot \begin{bmatrix} G_1(t) \\ G_2(t) \\ G_3(t) \\ G_4(t) \end{bmatrix} \quad (5.6)$$

Now, for a fixed t_l , $Q(s, t_l)$ is a curve because $G(t_l)$ is constant. Allowing t_l to take on some new value t_2 , then a new different curve $Q(s, t_2)$ is created, and so on until $t=l$. The sets of all such curves defines a surface.

5.3.2 Conversion of Pixel Representation to NURBS Representation

The pixel representation has the nice feature that it arranges the pixels on the x, y grid of the screen. This feature can be used to construct contours or curves that connect pixel center points along constant x and y grid addresses. Each of the planar contour points can be used as a control point to generate NURBS curves. Then two sets of equally spaced planar NURBS curves can be used to generate a NURBS surface. Alternatively, the contour points could be used to construct a triangular mesh for a smoother rendering of pixel objects to overcome the view dependent problem associated with ray-casting techniques [Hang and Oliver, 1995; Keppel, 1975].

Assuming contour with constant x is to be generated, the contour generation can proceed by first selecting a starting pixel that has the smallest Y coordinate value. Then it sequentially traverses all of the pixels with the same x address. The basic rule of the traversal is, from the current pixel point move to the next closest pixel point in the clockwise direction. There are two types of contour that can be generated from the pixels.

The first type is a simple contour, which occurs when each ray intersects with the object of concern at only two points (one in and one out). Several cases of pixel connection are illustrated in Figure 5.7. In case 1, the segment pixels overlap and pixel points are at the near side (smallest Z -values) of the view plane, thus the pixels are connected starting from the smallest y and moving to the next higher grid point in the y direction at each step until reaching the highest y address. Case 3 is similar to case 1, except it handles the pixel points at far pixel side. Thus pixels are connected starting from the highest y and moving to the next lower grid point in the y direction until reaching the lowest y address. In case 2, the pixel points are located on the same ray and y address.

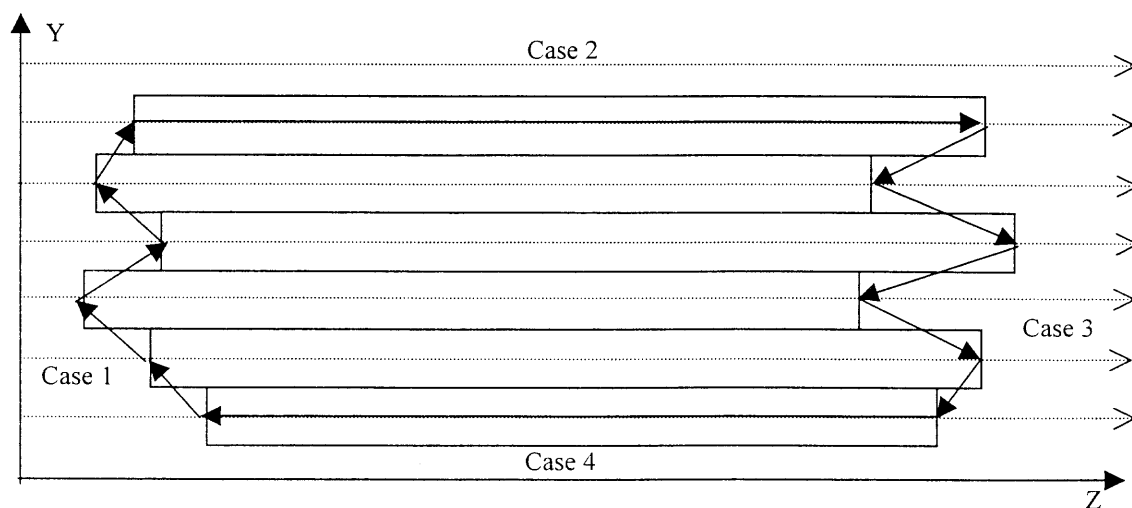


Figure 5.7 Cases of pixel point connections for constant x address

For this case, the near and far highest pixel point in the y address are connected. In case 4, the pixel points of the same y address are connected in a similar fashion.

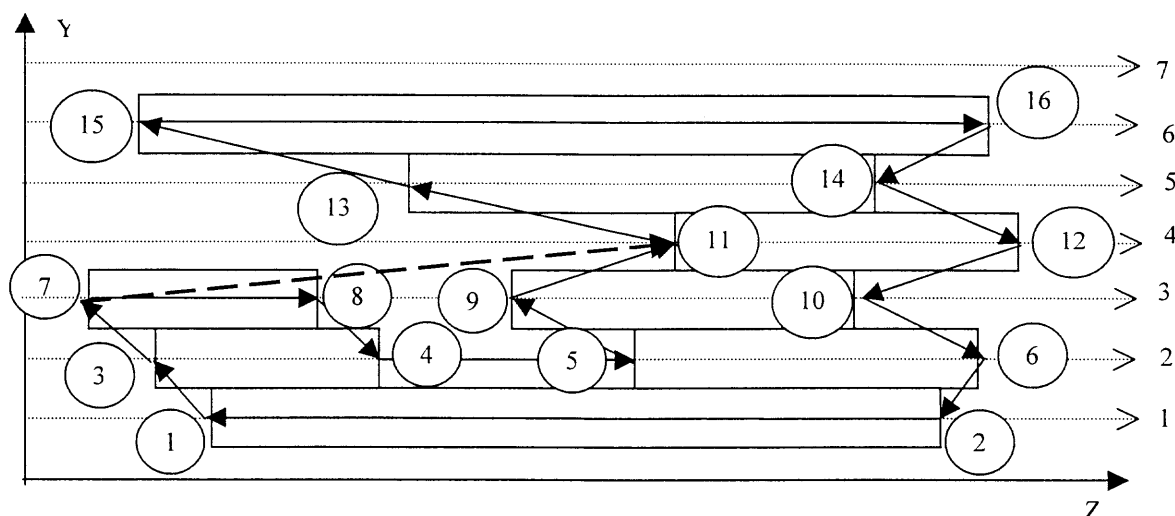


Figure 5.8 Cases of pixel point connections for curves

The second type of contour occurs when same of the rays intersect with the concerned object at more two points (i.e. four points or more). For this type of contour, we can not apply simple rules of connection as the first type of contour. Figure 5.8 shows this type of curve. If we apply the rules for the first type of contours, then the points 7 and 11 will be connected because they are on the near side of the view plane. This is obviously incorrect. The correct connection should be between the points 7 and 8.

To overcome this problem, a new general method has been developed for any contour type. The main idea of this method is to update the number of the depth values in order to keep the rule of connecting the nearest point first. The steps of the method are as follows:

1. Create a table with two columns, where the first column is the y-address of the pixels and the second is made up of the depth values, which are sorted from the smallest to largest Z-value in the screen coordinate system.

2. Select a start connection point that has the smallest depth value in the same pixel or constant y-address.
3. Find the smallest depth value point (new point) among the three points, which are the next grid point in the same y-direction, next higher point in the y-direction and the next lower point in the y-direction of the start point. If two values are equal, then select any point and call it a new point.
4. Connect the start point with the new point if they overlap, otherwise connect with the next point in the same y-address.
5. Delete the start point from the table and add it to the list of the contour curve.
6. If the number of the points in the table is larger than one, then go to step 3 after renaming the new point as the start point.
7. Connect the last point in the table with the first selected point to create a closed contour curve.

To illustrate the method, Figure 5.8 is used as an example.

- The initial table is created as shown in table 5.1.
- Point 1 is selected as the start. It has the smallest depth value of all pixels at its y-address. It has been marked in the table by a circle.
- Point 1 is connected with point 3, which it is the next higher grid point, because it has a smaller depth value than point 2, which has the same y-address as point 1, and there are no points in the lower grid.
- Next, point 1 is deleted from the table and point 3 becomes the new start point. Also table 5.1 is updated to table 5.2.

- The smallest depth value among the points 2,4 and 7 occurs at point 7. Therefore point 3 is connected to point 7.

Table 5.1 The depth values and y-addresses for Figure 5.8

Y-address	The depth values or Z-values sorted
1	①→2
2	3→4 5→6
3	7→8 9→10
4	11→12
5	13→14
6	15→16

- Point 7 becomes the new start point and point 3 is deleted from table.
- The smallest depth value among the points 8,11 and 4 occurs at point 8, so point 7 is connected to point 8.
- Point 8 becomes the new start point and point 7 is deleted from table.
- The smallest depth value among the points 9,11 and 4 occurs at point 4. Hence point 8 is connected to point 4.
- Point 4 becomes the start point and point 8 is deleted from table.
- The smallest depth value among the points 5,9 and 2 occurs at point 9. But the point 4 is connected to point 5, because the segment 9→10 does not overlap point 4.

Table 5.2 The depth values and y-addresses for Figure 5.8

Y-address	The depth values or Z-values sorted
1	2
2	3→4 5→6
3	7→8 9→10
4	11→12
5	13→14
6	15→16

- Point 5 becomes the new start point and point 4 is deleted from table.
- The smallest depth value among the points 6,9 and 2 occurs at point 9.

Therefore point 5 is connected to point 9.

The process is continued until all points have been connected correctly, as shown in Figure 5.8.

The result of the developed method is that the pixel objects are converted to contour curves. Then the contour curves can be connected to create surfaces that represent the objects. Several researchers have studied the problem of creating triangulated surfaces from convex and concave contours [Ekoule, et. Al, 19991; Meyers, et. al, 1992 and Keppel, 1975].

Our aim is to convert the object from pixel representation to NURBS representation, namely the NURBS surface representation. It is easy now to convert the contour curves to NURBS curves by using the equation (5.5), where the basis matrix can be found in the CAD/CAM text book [Zeid, 1990]. After the NURBS curves have been

created, then the NURBS surface is created by lofting or skinning techniques which most existing CAD/CAM packages support. For example, Pro/Engineer (a commercial CAD/CAM package) can import the contour curves generated from our VR system as open or closed section curves. Pro/Engineer constructs cubic B-Spline curves from the contour points. Also Pro/Engineer provides the user with several options to manipulate and analyze the curves. More details can be found from the Pro/Engineer manual.

A typical pixel representation of solid geometry can easily generate thousands of pixel points on the contour. However, not all the pixel points are necessary for the contours. For example, sequential pixel points that have the same normal vector or nearly collinear normals can be reduced to two end points. Thus the intermediate pixel points are eliminated by a culling process that checks sequential pixel points, of the same normal vector, against a linearity tolerance. The points that have been identified with the same normal vector are input to Pro/Engineer as a linear B-Spline curve (which is a line); other points are input as cubic B-Spline curves.

5.4 Analysis of the Design Model Accuracy

The desired outputs of the VR system are freeform surfaces and objects describing the geometry of the designed part suitable for production in the real world. The details of the VR environment have been described in detail in terms of the hardware and software. In order to solve the real-time issues in our system, several approximations have been made which affect the design model accuracy.

To analyze the accuracy of the design model and develop a method to control accuracy, we need to understand and analyze the accuracy of each component separately

in the VR system. Thereby we shall be able to understand to what extent each component affects the overall accuracy of the design model.

There are three main sources of inaccuracy in the design model, which are:

1. The uncertainty and accuracy of the VR hardware.
2. The accuracy of representing the design model in our solid engine.
3. The stability of the user movements.

Studying and understanding the stability of the user movements is very complicated because it is related to many factors such as the characteristics of the user, the motion control system of the human, the skills of the user, etc. The force feedback tends to reduce user instability. We shall not investigate user movement stability, and so in what follows we assume that no inaccuracies are caused by human motion instabilities.

In our analysis, we concentrated on the first two main factors affecting the accuracy of the design model. First we analyzed the VR hardware to determine the uncertainty and accuracy of the motion tracker device that is used to track the virtual tool motion. Next that the accuracy of the solid engine of the VR system was analyzed in terms of the representation of the virtual objects.

5.4.1 Uncertainty and Accuracy of the VR Hardware

The VR hardware input device in our VR system is a motion tracker. The type of the device we used is a magnetic type called the Flock of Birds (FOB). The FOB is a six degree-of-freedom measuring device from Ascension Technology Corporation. The FOB determines position and orientation by transmitting a pulsed DC magnetic field that is simultaneously measured by the sensor in the Flock. From the measured magnetic field

characteristics, the sensor computes the position and orientation and makes this information available to the host computer.

Table 5.3 shows the technical specifications of the FOB, which are obtained from the Ascension Technology Corporation web page [<http://www.ascension-tech.com>].

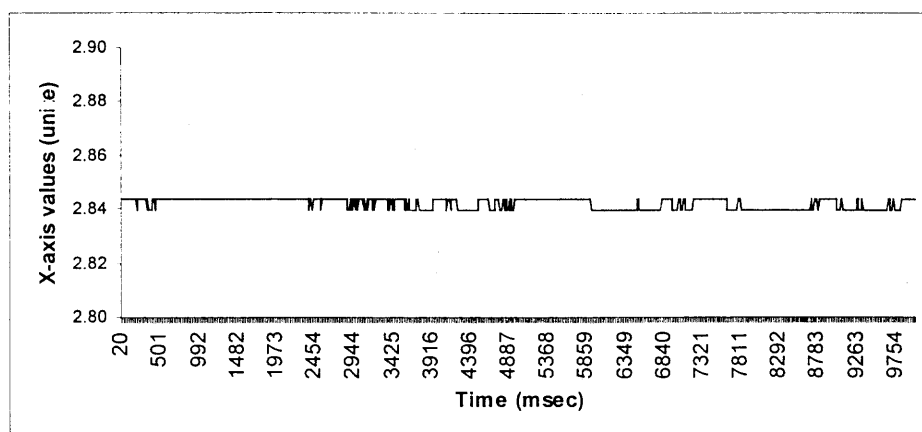
Table 5.3 The technical specifications of the FOB

Translation range	$\pm 4'$ ($\pm 10'$ optional) in any direction
Angular range	$\pm 180^\circ$ Azimuth & Roll $\pm 90^\circ$ Elevation
Static Accuracy Position	0.07 inch RMS
Static Accuracy Orientation	0.5° RMS
Static Resolution Position	0.02 inch @ 12"
Static Resolution Orientation	0.1° @ 12"
Update rate	Up to 144 measurements/second
Outputs	X,Y,Z positional coordinates and orientation angles, rotation matrix, quaternions

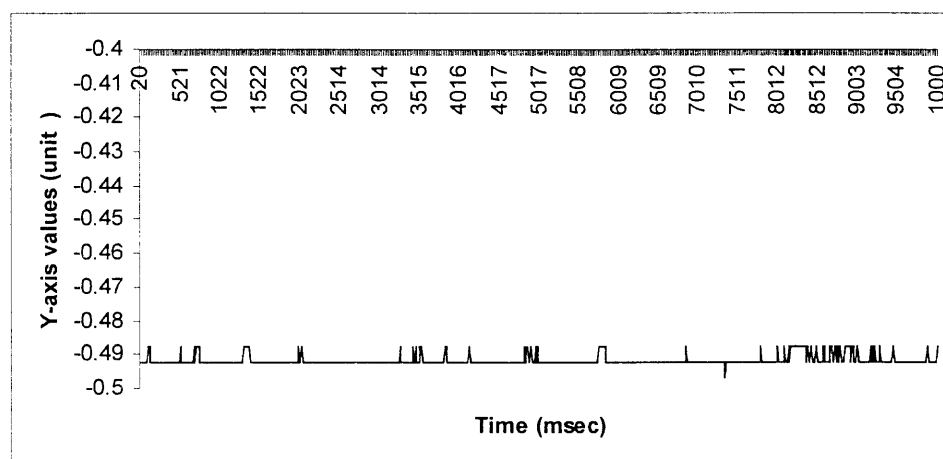
An experimental setup was used to test and measure the specifications of the FOB tracker device. The first test is to measure the uncertainty of the FOB readings. The test is as follows:

1. Run the FOB device.
2. Run the computer software to configure the FOB device.
3. Place the sensor in the workspace of the transmitter.
4. Read and record the position and orientation data every 1ms.
5. Repeat step 4 at different positions in the workspace.
6. Quit from the software and stop the FOB device.

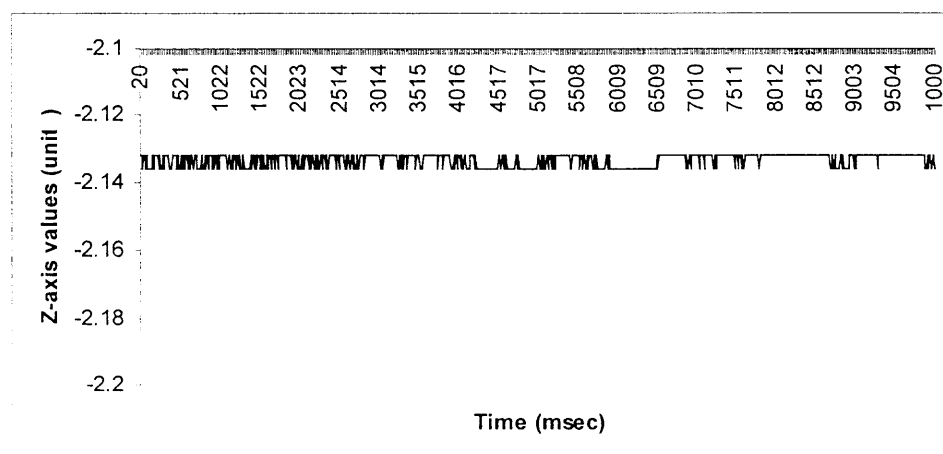
The results of the uncertainty test are shown in Figure 5.9 for displacement and Figure 5.10 for angles. It is clear that the variation of the values of displacement and angles is in



(a)

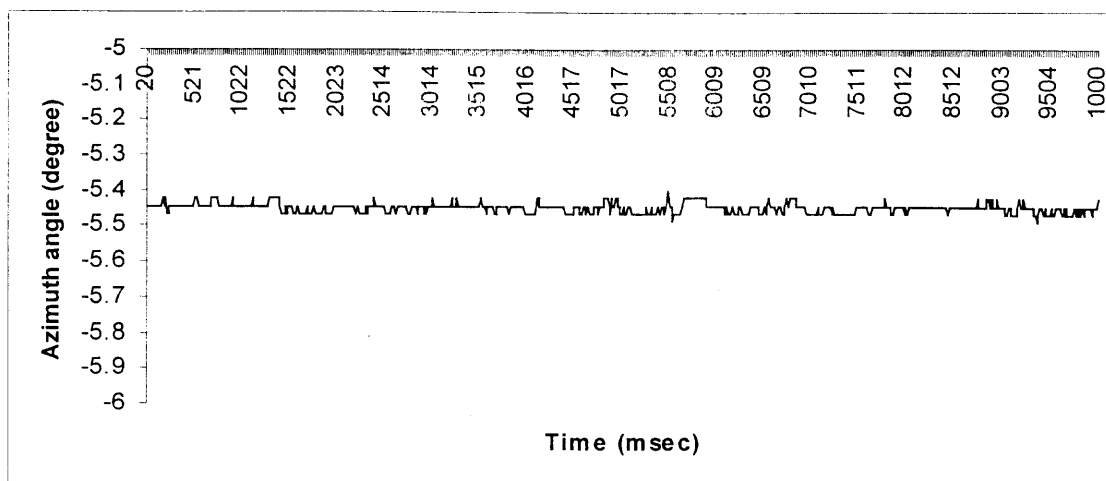


(b)

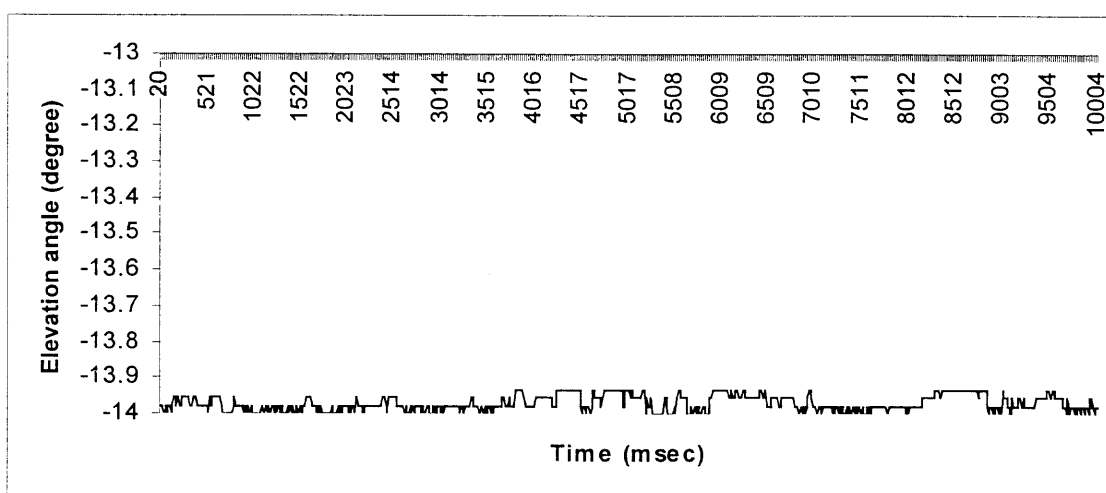


(c)

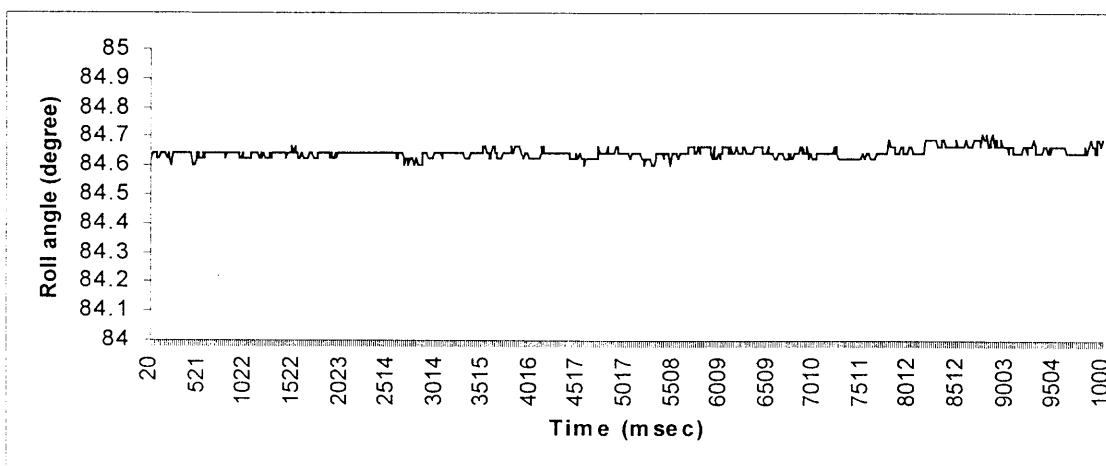
Figure 5.9 The position tests for the motion tracker in: a) X-direction, b) Y-direction and c) Z-direction positions



(a)



(b)



(c)

Figure 5.10 The orientation tests for the motion tracker: a) Azimuth, b) Elevation, and c) Roll angles

the range of 0.02 inch and 0.1°, respectively. The results of the test are similar to the manufacturer's specifications.

The second test is to determine the accuracy of the FOB device. The steps of the test are as follows:

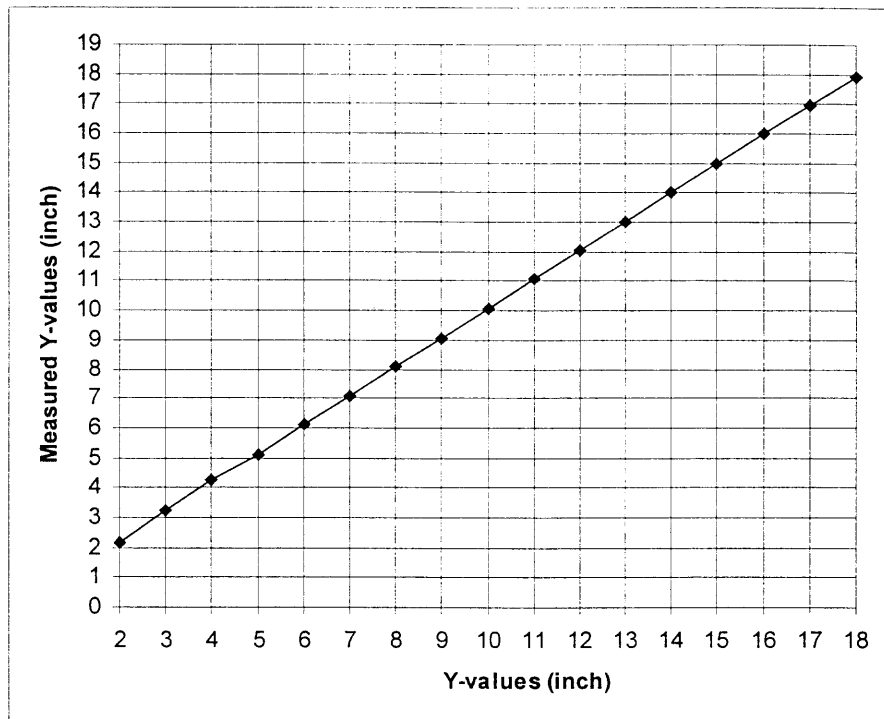
1. Place the transmitter of the FOB device at the origin of the board.
2. Run the FOB device.
3. Run the computer software to configure the FOB.
4. Place the sensor at the origin of the board (x=0, y=0).
5. Read and record the position and orientation data at every increment one step in the y-direction.
6. Repeat step 4.
7. Quit from the software and stop the FOB.

The results of the accuracy test are shown in Figure 5.11. The absolute percentage error of the measured values has been computed as follows:

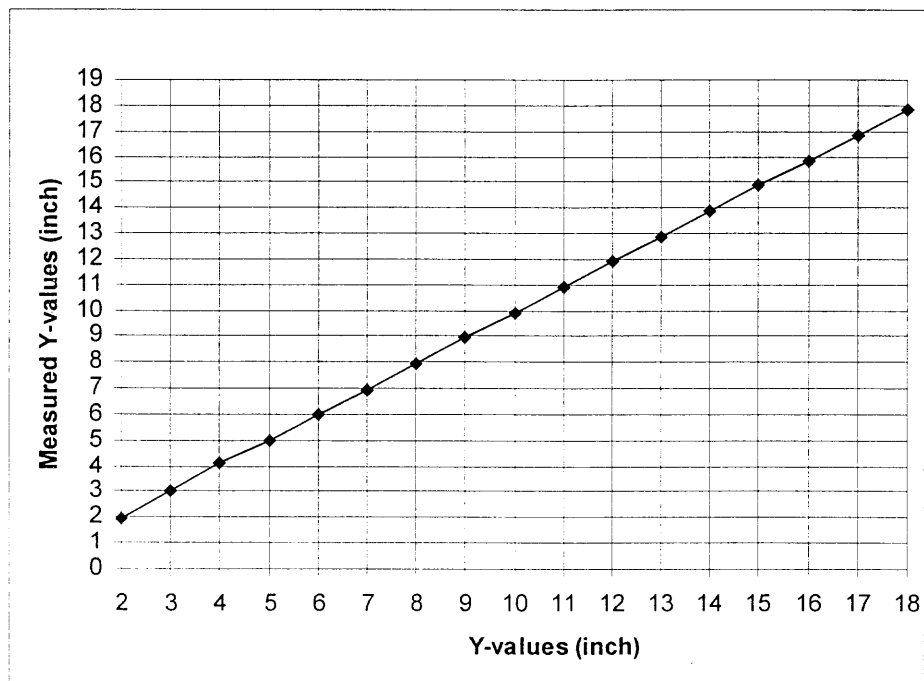
$$\text{Absolute (\% error)} = \frac{|\text{Ideal Value} - \text{Measured Value}|}{\text{Ideal value}} \times 100\%$$

Figure 5.11 shows the absolute percentage error of the measurements. The absolute percentage error is less than 3.5%, which is within the manufactures specification range.

The results of these tests can provide us with a good idea of the uncertainty and accuracy of the motion tracker device that has been used. The uncertainty and accuracy of the motion tracker device needs to be considered in our analysis of the design model accuracy. If the operator sweeps the virtual tools over a known object such as straight edge or disk, It is necessary to compensate for the difference between the real world geometry and measured virtual geometry.



(a)



(b)

Figure 5.11 The measured Y-values from motion tracker vs. the ideal Y-values for (a) Trial one (b) Trial two

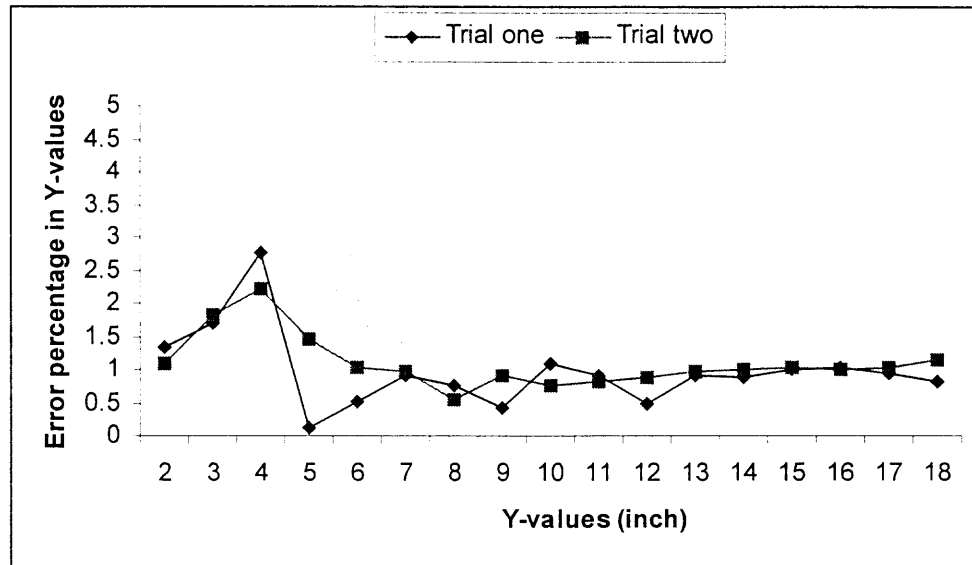


Figure 5.12 Comparison of the error percentage in Y-values in trials one and two

5.4.2 Accuracy of the Solid Modeling Engine

The solid modeling engine consists of two main components, which are the SDE method for swept volume computation and representation, and the ray-casting method for Boolean operations.

The accuracy of the swept volume is determined by two main factors, which are the accuracy of the representation of the tool geometry and the accuracy of the interpolations of the virtual tool trajectory (time step). We assume that the virtual tool has been triangulated linearly with triangles of diameter $\leq m^{-1}$ for some integer $m \gg 1$. Also let us assume that the interpolations of the virtual tool trajectory employ a time step for computing the swept volume $\leq m^{-1}$. Then the global error in approximating the candidate surface is $\leq m^{-2}$ [Blackmore, et. al, 1999]. The accuracy of the ray-casting method or pixel representation of any solid object is determined by the pixel plane orientation and the pixel size. To simplify the pixel operations and accuracy analysis, in

this implementation, the width and length of the pixel are assumed equal, and set equal to a value (w). Two methods are provided for specification of the pixel coordinate system in order to determine the view plane. The first is via interaction selection where the user chooses the faces that are exposed by using the virtual control menu options. The second approach automates this procedure by calculating the pixel view plane orientation that produces the maximum projected area of the design surfaces on the pixel plane.

The pixel representation is the key point for controlling the accuracy of the design model because the workpiece and swept volume must be represented in pixel form in order to perform the Boolean operations. If the user defines the accuracy or tolerance of the design model to be T , then the important question is "How to adjust the VR system to meet this tolerance?". We have developed a method to answer this question as described in the follows.

The size of the each pixel size is calculated from the user-specified approximation tolerance T , in our VR environment for freeform surface and object creation, where the size of the virtual tool determines the minimum feature size of the resulting part. Therefore, as shown in Figure 5.13, the pixel size is computed relative to the tool radius R as

$$T^2 + \left(R - \frac{w}{2}\right)^2 \cong R^2 \quad (5.7)$$

The ratio of T to R (T/R) characterizes the relative accuracy of the pixel representation.

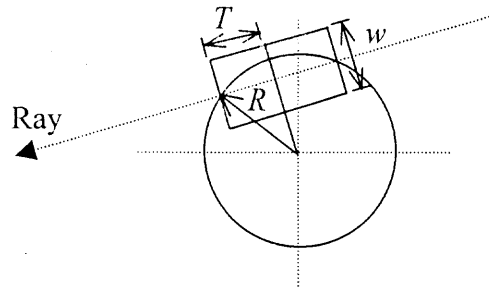


Figure 5.13 Approximation error of virtual tool in pixel representation

For a given tool size and tolerance then the size of the pixel is determined by the equation (5.7). Also from this information, the triangle diameter ($\leq m^{-l}$) of the linear triangulation for the virtual tool for swept volume computation is determined by the ratio of R to w (R/w), where $(R/w) \gg 1$.

The discretization of the virtual tool trajectory is a key factor in the VR system application because it is related to several aspects of the process, which are the time step in the swept volume computation, the measurement rate of the motion tracker, the speed of the user and the solid modeling engine computation time (real-time issue).

Let the start and end cutter location (CL) points of a tool motion be given by P_1 and P_2 , respectively, and the corresponding unit vectors along the tool axes by u_1 and u_2 . Transforming the CL points and axes into the pixel coordinate system yields P_1^{λ} , P_2^{λ} , u_1^{λ} and u_2^{λ} , respectively, as shown in Figure 5.14.

To model the swept volume using the SDE method, the time step size (Δt) is $\leq m^{-l}$. This means the step time should be less than or equal to (w/R) , $\Delta t \leq (w/R)$. But the time step of the CL data is determined by the measurement rate of the motion tracker device, which is constant and equal to *100 Frame/second or 100 data/second*. Hence the number of data (N) needed to represent the swept volume with a global error in approximating the surface of order $= m^{-2}$, is larger than or equal to $100 \times \Delta t$ ($100 \times w/R$).

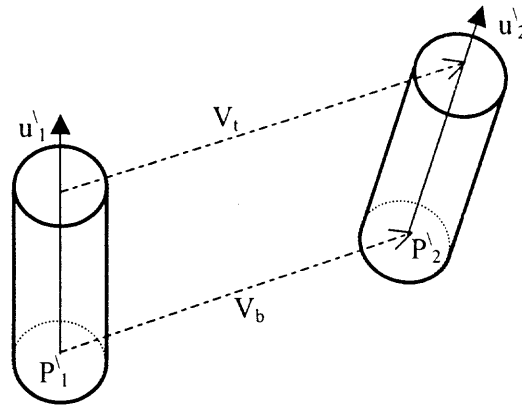


Figure 5.14 The tool motion

On the other hand, the number of data between the CL data points for pixel representation is given by:

$$N = \frac{\max(|v_t^x|, |v_t^y|, |v_t^z|, |v_b^x|, |v_b^y|, |v_b^z|)}{w} \quad (5.8)$$

where V_t and V_b are the linear sweeping vectors of the tool top and bottom center points in the pixel coordinate system (as shown in Figure 5.14) and $v_t^x, v_t^y, v_t^z, v_b^x, v_b^y$ and v_b^z are defined to be the x , y , and z -coordinates of the following points

$$\begin{aligned} v_b &= P_2 - P_1 \\ v_t &= v_b + L(u_2 - u_1) \end{aligned} \quad (5.9)$$

where L is the length of the tool.

The maximum speed of the user's hand (S) in the virtual world to meet the pixel representation requirement is given by:

$$S = \frac{\max(|v_t^x|, |v_t^y|, |v_t^z|, |v_b^x|, |v_b^y|, |v_b^z|)}{\Delta t} \quad (5.10)$$

where t is the time.

The computation time needed to compute the swept volume, perform the Boolean operation, and update the image, is used to determine the speed of the user's hand. The VR system gives the user a feedback or warning to adjust hand speed movement in order to meet the computation time requirements and design model accuracy.

Figure 5.15 shows the flow diagram of the feedback system to the user in order to reduce hand movement speed. First the user inputs the design model accuracy requirement and tool geometry (length and radius). Then developed software computes the pixel size and triangulates the virtual tool to compute the swept volume. Then the user starts to carve the design model and move the tool at a certain speed (S_{actual}). The computation time is calculated for swept volume and Boolean operation. From the computation time and the CL data we compute the ideal user speed (S_{ideal}). If the user's ideal speed (S_{ideal}) is larger than or equal to the user actual speed (S_{actual}) then the user can continue the process at the same speed; otherwise the VR system displays a warning message to the user to decrease the hand movement speed or else suffer a decrease in accuracy for the design model. Our system allows a hand speed of approximately 10 unit/sec, while producing accuracy (tolerance) of approximately 0.02 units.

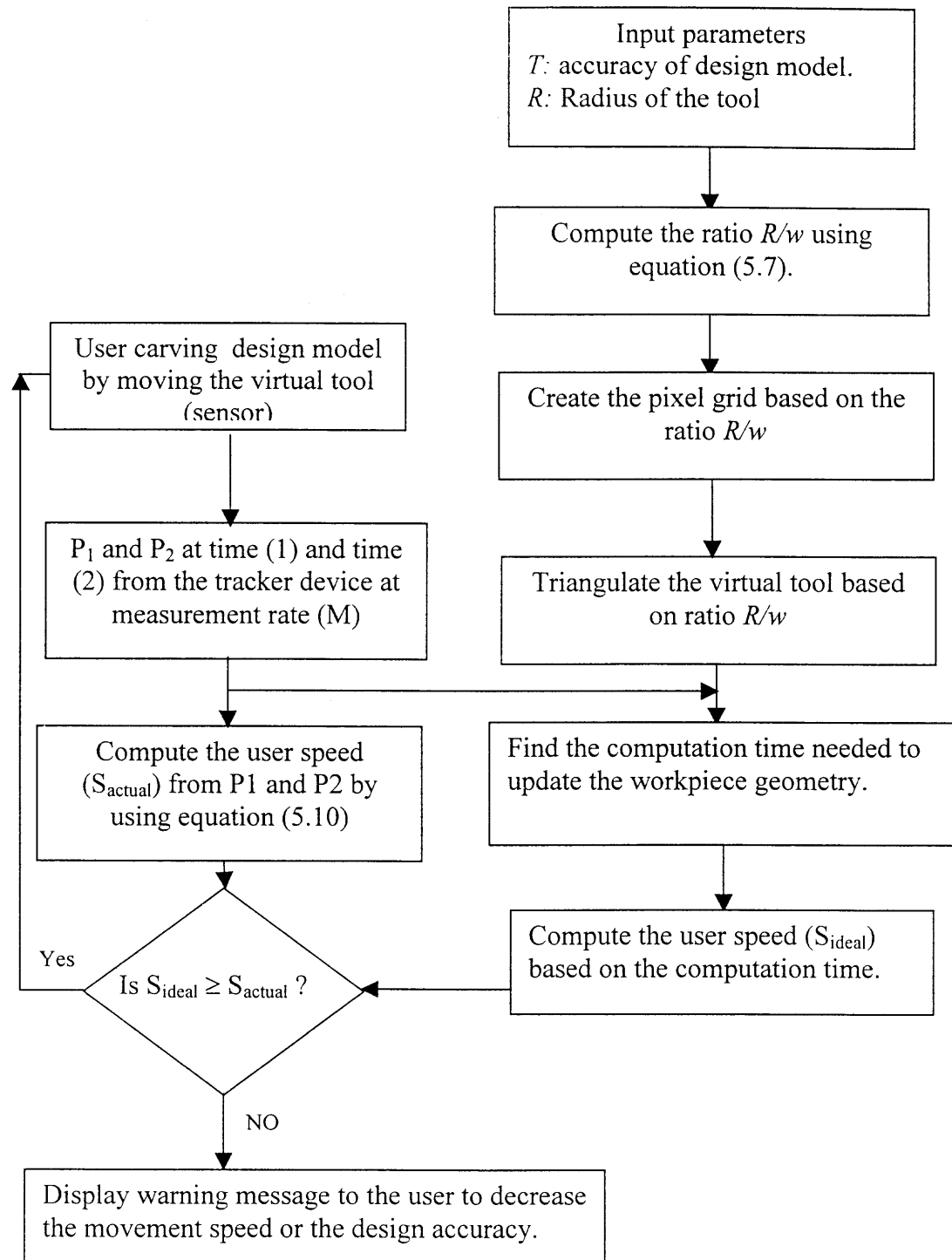


Figure 5. 15 The feedback system to adjust the user hand speed

5.5 Demonstration Examples for the VR system

The following examples illustrate how to use our VR environment to generate the freeform surfaces and input them into commercial CAD/CAM packages (Pro/Engineer). The virtual tool that has been used makes it simple to create complex objects. Also the virtual tool was used to demonstrate the editing features of the Virtual Reality for the creation of freeform surfaces and objects.

5.5.1 Example 1

In this first example, we created the logo mold for our school (New Jersey Institute of Technology). Figure 5.16 shows the results of the designing process. The letter "N" illustrates the capability of the VR system to generate straight lines at different angles to one another. Also it is clear that the user needs to plan a sequence of hand movements in order to create the letter "N" in one cut. In this test the user begins from the lower left

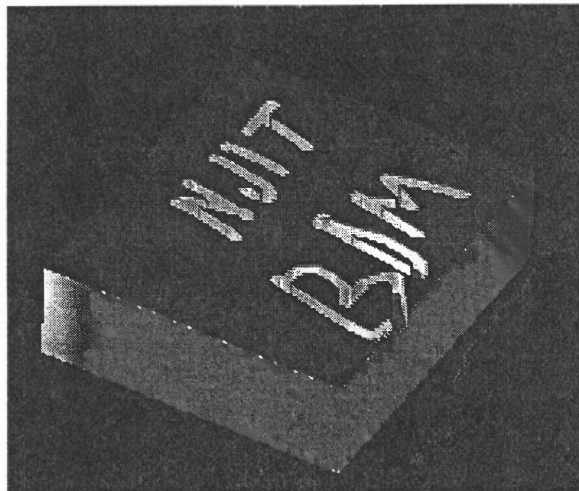


Figure 5.16 The NJIT Logo created by our VR system

side of the letter "N" and continues until reaching the upper right side. This is a very natural tool trajectory for creating the letter "N" in one continuous motion. Creating the

Letter "J" illustrates the capability of the VR system to create smooth curves. The letter "T" demonstrates the capability of the VR system to create intersecting curves. Of course the VR system makes it easy to create the straight line segment "I". The first letters of my name (Bilal Maiteh), which are the letters "B" and "M", demonstrate the capability of the our system to create a combination of straight lines and curves, and this is most evident in the letter "B".

The example is very simple but it gives a very good idea a bout the type of curves that can be generated-actually since the workpiece is three-dimensional, the curves are more accurately described as surfaces. Also the example shows how the plan for generating an object can be formulated by the user prior to operating the VR environment. For example, the user needs to determine the size of the tool required to create the letters within the size of the workpiece. The plan for the carving of the object in this example actually minimizes the time needed to complete the design. The tool motion starts at the left lower corner of the letter "N" and finishes at the upper right corner. Then the virtual tool is removed from the workpiece and begins curving again at the lower left corner of the letter "J" and finishes at the upper right corner. The process continues until the whole design part is finished. This process may be repeated several times in order to improve the look of the design, but that is one of the main advantages of our system. The designer or user can manipulate the objects often as desired with a minimal cost.

5.5.2 Example 2

The second example is the mouse shape shown below. Figure 5.17 shows the VR

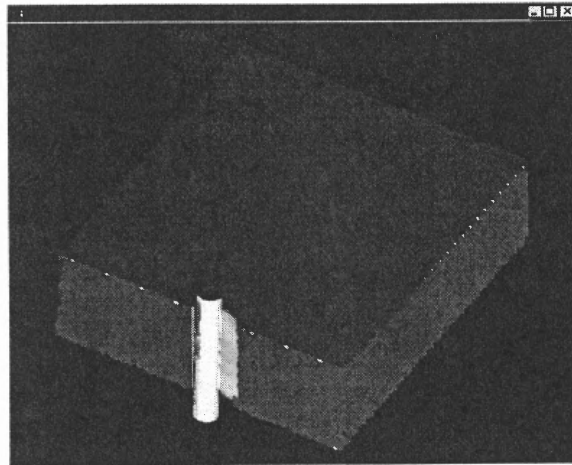


Figure 5.17 The start step in designing the mouse

environment just after initiation of the design process. The tool diameter selected is 1mm and the length is larger than the virtual workpiece thickness. The accuracy of the design part, as determined by the ratio of T to R , is equal 0.1 and corresponds to w/R equal 0.01. The number of points or triangles representing the virtual tool is determined on the basis

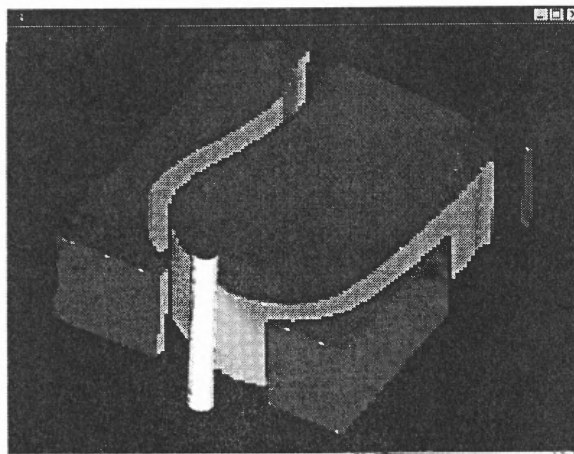


Figure 5.18 The second step is creating the freeform surface of the mouse

of the w/R ratio and the time steps for the tool movements are determined from the motion tracker device measurement rate. Figure 5.18 shows the shape of the design model after one complete sweep of the cutting tool by the user to create the freeform surface of the mouse. The user can repeat this process several times until the desired shape is obtained.

The next step in designing the mouse is to remove the extra material around the mouse. In this stage the user can increase the cutting speed or increase the virtual tool diameter since accuracy is not crucial. Figure 5.19 (a) shows the user removing the extra material by keeping the same tool diameter and increasing hand movement speed. Figure 5.19 (b) shows the final results of the mouse after removing the extra material.

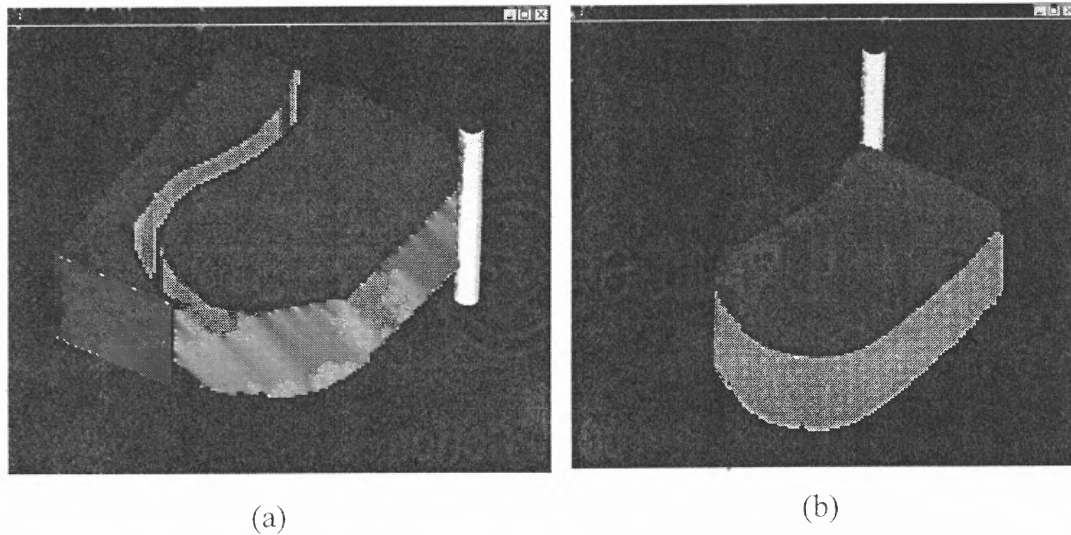


Figure 5.19 Final step in mouse design: (a) removing the extra material around design model (b) the final design model of the mouse

5.5.3 Example 3

The last example we present here is a model of the Mickey Mouse's head. The reason for selecting this example is that it has many features that are difficult to model using existing CAD packages. Figure 5. 20 shows the design model of the Mickey Mouse after one complete sweep of the virtual tool by the user.

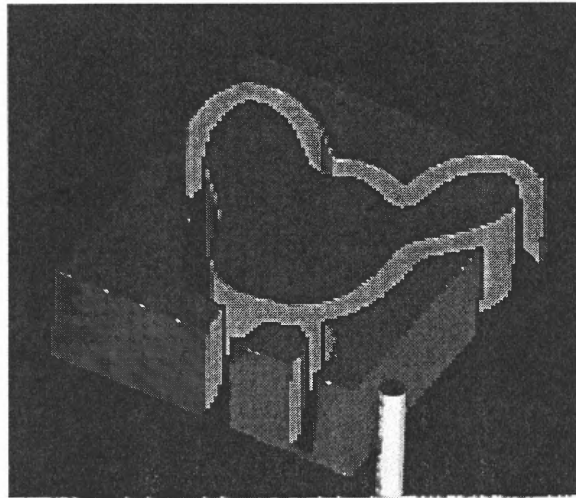
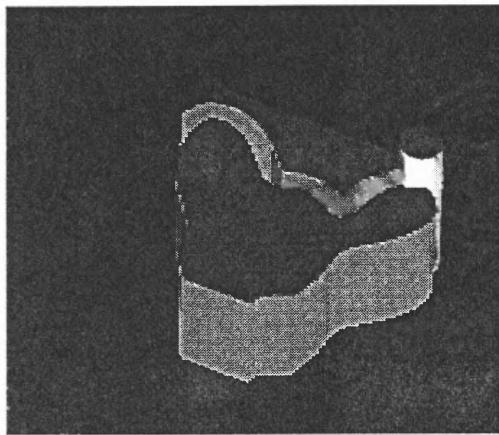
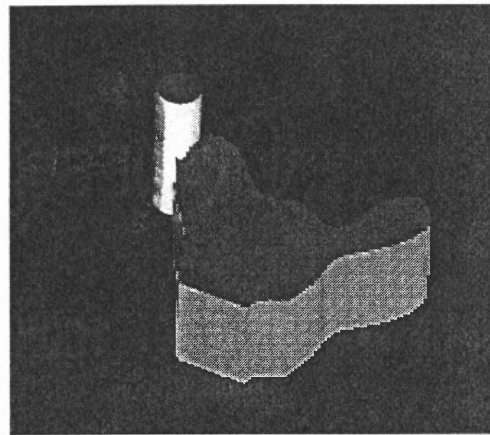


Figure 5.20 First cut of Mickey Mouse



(a)



(b)

Figure 5.21 The final stages of Mickey model (a) Removing the extra material with a larger tool (b) the final design model of Mickey

Next a larger sized tool is used to remove the extra material around Mickey's head. Figure 5. 21(a) shows the removal process with the large tool and the final shape of Mickey is shown in Figure 5.21(b).

After the above steps, the contour lines of Mickey's head are saved in NURBS representation and then saved in a file format which can be input to existing CAD/CAM packages. The Mickey head model has been input to Pro/Engineer, which is a commercial CAD/CAM package from Parametric Technology Corporation. Figure 5.22 shows the B-Spline curves of Mickey after being imported to Pro/Engineer.

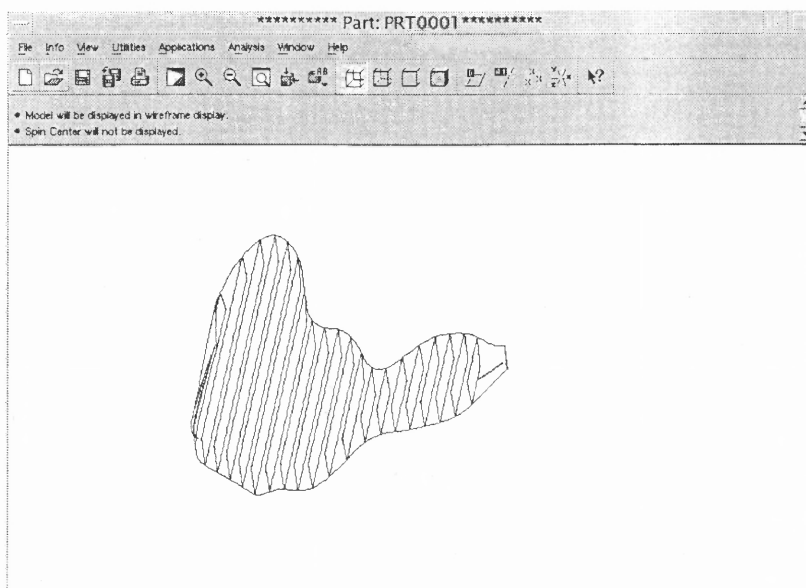


Figure 5.22 The B-spline curves of the Mickey face model after importing into CAD/CAM package

Figure 5. 23 shows the solid modeling of Mickey's head viewed from different directions in Pro/Engineer.

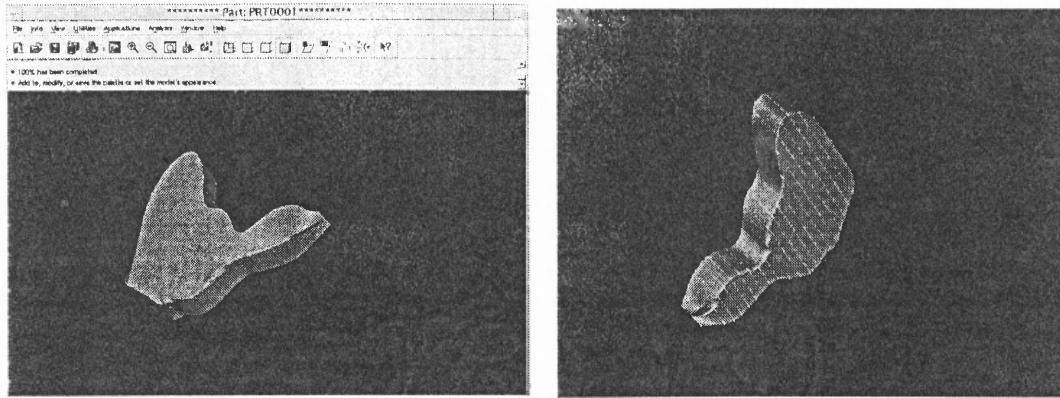


Figure 5.23 The solid model of the Mickey head in two different directions after being imported to Pro/Engineer as NURBS representation

CHAPTER 6

INTERACTIVE BOOLEAN OPERATIONS FOR DESIGNING 3D SOLIDS IN A VIRTUAL REALITY ENVIRONMENT

6.1 Introduction

In this research, we presented a new approach that uses Virtual Reality to create 3D solids and freeform surfaces for conceptual design. The solid modeling engine that has been developed to meet the VR requirements, uses the SDE method for generating swept volume and ray-casting to perform Boolean operations. A prototype of the VR system has been developed and built-this prototype effectively demonstrates the basic capabilities and usefulness of our new approach.

One important component of our VR system that enables real time computations in the design process, is a new algorithm for Boolean subtraction based on a combination of the SDE and ray-casting techniques. Existing approaches for Boolean operations in CSG and B-rep do not support interactive conceptual design and require expensive special-purpose hardware that is not easily available. In this chapter, existing approaches for Boolean operations are reviewed and discussed. Then our new algorithm is described and its computational complexity is analyzed.

6.2 Background

A design process starts with a conceptual design and progresses by iterative refinement stages until meeting the design goal. In geometric design, the initial conceptual design phase is one of the most difficult to computerize. Other stages, such as detailed geometric design and physical analysis, are already supported in current modeling systems. Conceptual design is rarely supported [Smithers, 1989].

Most conceptual design is currently done using pen and paper. When designing 3-D models, the great potential advantage of computer graphics is obvious-it enables designers to inspect their models from different viewpoints and at different scales and make fast modifications. We are interested in the direct conceptual design of 3-D models.

In Chapter 4, we reviewed the major types of representation approach to solid modeling packages, which are Constructive Solid Modeling (CSG), Boundary Representation (B-rep), Sweep Representation and Spatial-Partitioning Representations. These solid modeling packages support Boolean operations. In this section, we discuss in detail the Boolean operation approaches that are used in solid modeling.

No matter how we represent objects, we would like to be able to combine them in order to make new ones. One of the most intuitive and popular methods for combining objects is by Boolean set operations, such as union, difference, and intersection, as shown in Figure 6.1. These are the 3D equivalents of the familiar 2D Boolean operations. There are two types of Boolean set operations, which are ordinary Boolean set operations and Regularized Boolean set operations [Requicha, 1977]. The difference between the two types is that the Regularized Boolean set operations always yield solids, but the ordinary Boolean set operations do not necessary yield solids. For example, ordinary Boolean set intersections of cubes may yield a solid or a plane or a line or a point or a null object as shown in Figure 6.1 from (a) through (e). But the results of the Regularized Boolean set intersections are solids or the null set as shown Figure 6.1 (a) and (e); the cases (b) through (d) are defined to be empty when regularized. The Regularized Boolean set operations are often used in solid modeling.

The Regularized Boolean set operations method for the spatial-partitioning representations has been discussed in Chapter 4. Regularized Boolean operations for the polygonal decompositions used in Boundary Representation will be discussed in this chapter. In general, Constructive Solid Modeling and Sweep Representations are converted to spatial-partitioning or a polygonal representation to display the results of the Regularized Boolean set operations.

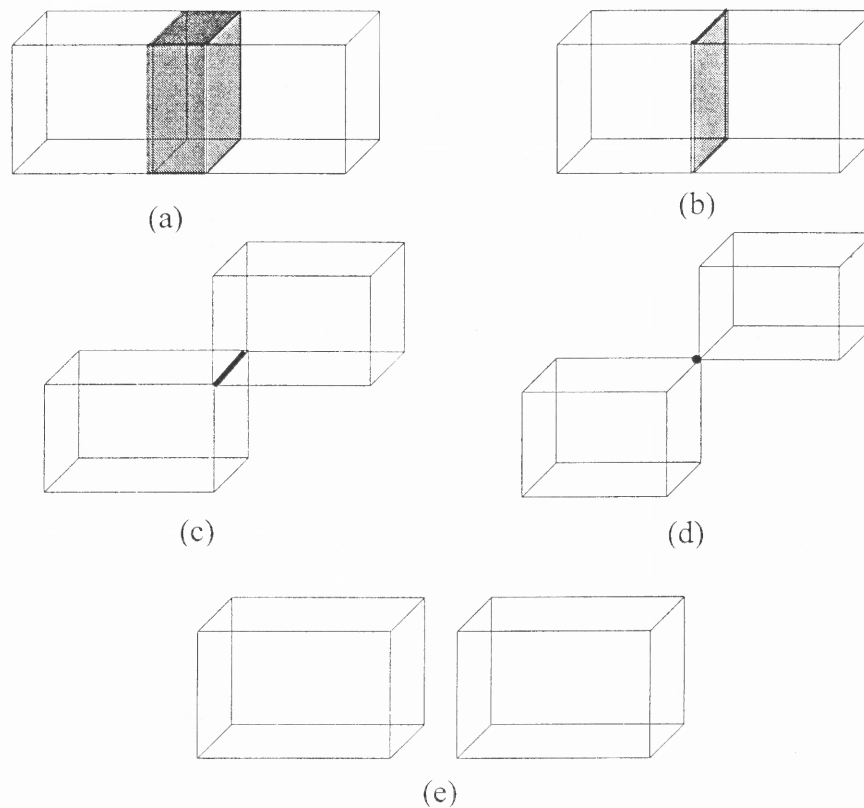


Figure 6.1 The ordinary Boolean intersections of two cubes may produce (a) a solid, (b) a plane, (c) a line, (d) a point, or (e) the null set

Most of the virtual object models used in our VR environment (and other VR systems as well) are polygonal models because these models have characteristics conducive to real-time computation. The Regularized Boolean set operations on polygon

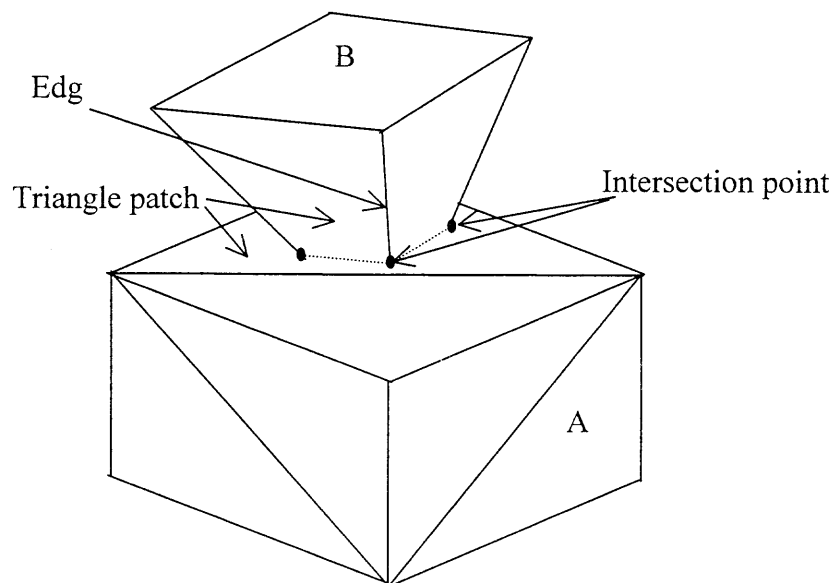
models has been extensively studied in the field of CAD (Computer Aided design). Miller [1987] and Sarraga [1983] discuss algorithms that determine the intersections between quadric surfaces. Algorithms for combining polyhedral objects are presented by Laid [1986] and Toriya [1991].

The only operation needed to compute the geometry of the virtual design models generated from our VR environment for the creation of the freeform surfaces and objects is the intersection between two polyhedra. Therefore we confine our discussion to the intersection operations. The main steps of the implementation of Boolean operations (intersection) described by Laid and Toriya and implemented in CAD/CAM systems are as follows (Figure6.2):

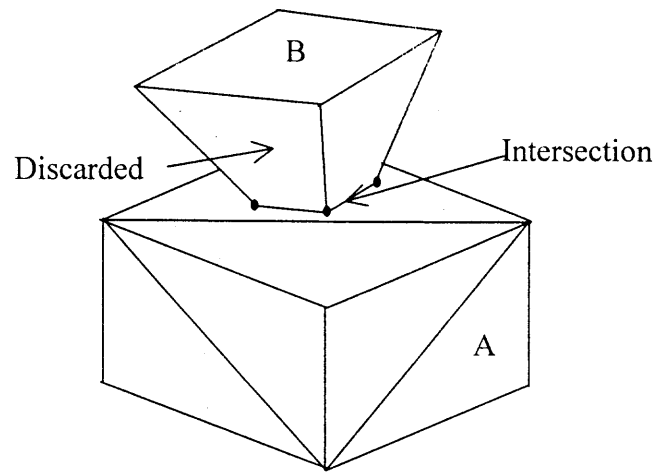
1. Solid models are all converted into a set of triangular boundary patches with normal directions indicating the external direction. Three nodes or three edges represent each patch.
2. Intersections of polygons: Let A and B represent the object of interest to be cut (virtual workpiece) and the cutter (Virtual cutter tool), respectively. The mutual intersections of the triangles belonging to A and those to B are checked through interaction of triangles belonging to A or B and edges to B or A, respectively. When duplications appear, this check is skipped. As long as interactions are found, we compute intersection points.
3. Intersection lines: The intersecting lines are those connecting the intersection points computed above. The intersection points are listed as new nodes. Also new edges are generated by connecting the new nodes in the plane of an existing triangular face. In

this manner new polygons corresponding to the cut surface are generated. Some of them may be polygons other than triangles at this step.

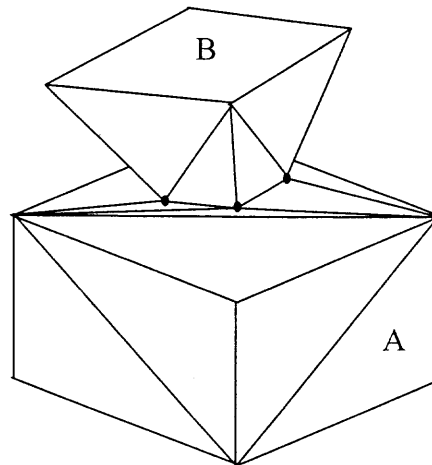
4. Decomposition into triangles: The polygons other than triangles generated above are decomposed into triangular patches. This decomposition is performed by adding edges to connect nodes.
5. Elimination of redundant elements: Some parts are to be discarded after cutting and others remain. Each node on the intersecting lines belonging to the object A and not in the interior of the tool B is marked. All of those nodes that are unmarked are discarded.



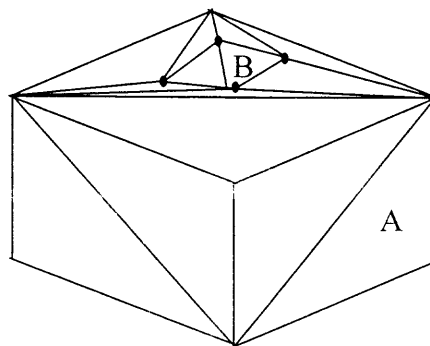
Step 1 & 2: Triangulation and intersection points



Step 3: Intersection line



Step 4: Triangle patches



Step 5: Delete discard

Figure 6.2 Steps of Boolean operation

6.3 Interactive Boolean Operations Approach

We developed a VR environment for the creation of freeform surfaces and objects. One of the major objectives of our VR approach is to create virtual design models with an accuracy equal to the approximation accuracy of the SDE method at a computational cost (time) roughly the same as that of the standard SDE algorithm. In order to develop this method, we utilized the natural features of the SDE method for computing and representing the swept volume.

The approach described in the previous section has main two steps: 1) Triangulate the objects created by the intersection of two polyhedra 2) Classify the elements of the triangulation after intersection to determine whether or not they lie within the interiors of the solid objects. The method that we have developed has the following ordered pair steps: 1) Classify the vertices of one object relative to the other to determine whether they lie inside, outside or on the boundary and then add the intersection vertices to obtain a new set of vertices 2) Construct an efficient triangulation of the intersected object using the new set of vertices. The main steps for implementing the latter method for Boolean operations (intersection) is as follows (Figure 6.3):

1. Solid models are all converted into a set of triangular patches with normal directions indicating the external direction. Three nodes or three edges represent each patch.
2. Intersections of Polygons: Let A and B represent the object of interest to be cut (virtual workpiece) and the cutter (Virtual cutter tool), respectively. The mutual intersections of the triangles belonging to A and those to B are checked through interaction of triangles belonging to A or B and edges to B or

A, respectively. As long as interactions are found, we compute intersection points. This step is described in detail in section 6.4.

3. Intersection lines: The intersecting lines are those connecting the intersecting points computed above. The intersection points are listed as new nodes on the edge that intersect the object. Also new edges are generated by connecting the new nodes in the same planes of triangular faces. In this manner new polygons

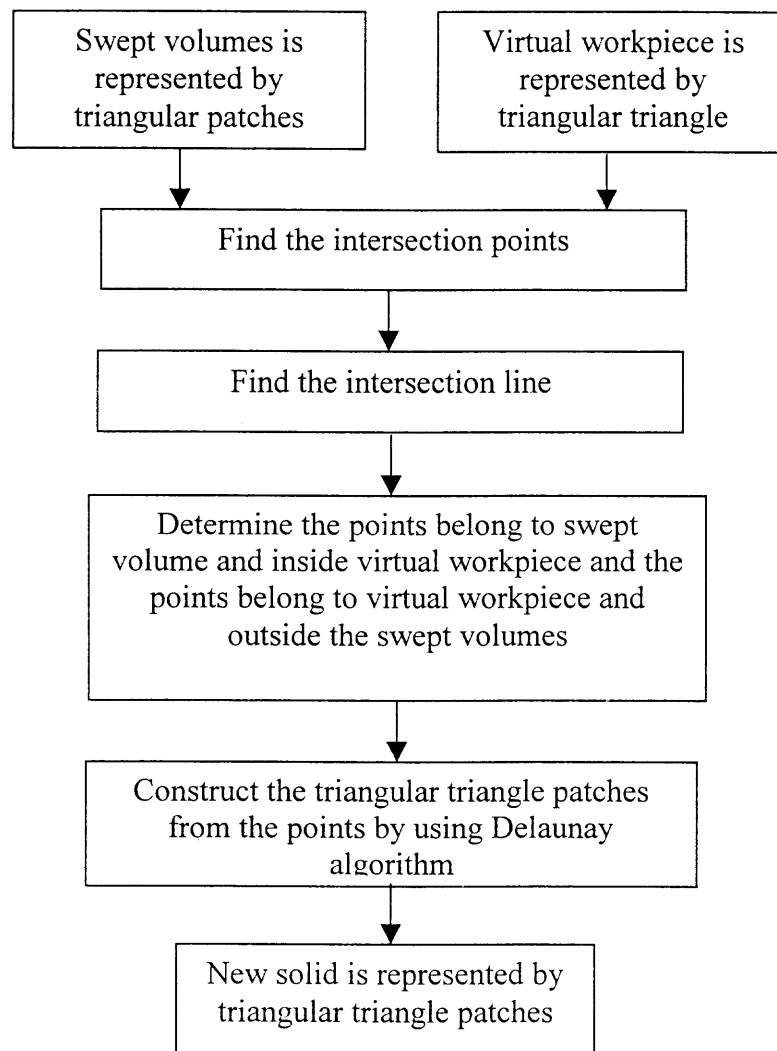


Figure 6.3 Steps of the interactive Boolean operations method

corresponding to the cut surface are generated. Some of them may be polygons other than triangles at this stage.

4. Elimination of unnecessary elements: Some vertices are to be discarded after cutting. Each vertex point belongs to A and inside B, is discarded. Also each vertex point that belonging to B and is outside A is discarded. This step is described in section 6.4.2.
5. Composition into Triangles: construct the triangular patches from the remaining vertices using the Delaunay algorithm. This step is described in section 6.4.1.

The method developed above is more efficient when it is integrated with the SDE method to compute and represent the solids that result from the Boolean operations. The details of this integration are described in the next section.

6.4 The Interactive Boolean Operations and SDE Method for Modeling the Virtual Reality 3D Solids

As we described in Chapter 4, the SDE method generates a set of points that are used to construct the triangulation of swept volume boundaries. The SDE method can compute the swept volume of a moving object with good accuracy. It provides a boundary representation for the swept volume. In this method, the sweep differential equation representing the velocity field is obtained by taking the time derivative of the motion equation which includes both translation and rotation. The boundary of an object is partitioned (at any time) into ingress, egress, or grazing points. To determine whether a point is an ingress, egress, or grazing point, one can simply compute and interrogate the inner product(s) of the velocity vector and the outward normal vector(s) at that point. We

have shown that the grazing points on the boundary of a three-dimensional object generally form a curve and that the boundary of the swept volume of an object is formed essentially by joining the grazing curves along the sweep. Thus, the SDE method creates curves from the grazing points. This observation can be used to efficiently integrate the SDE method with interactive Boolean operations as follows:

1. Convert the workpiece to a triangulated boundary representation: the initial workpiece is a simple rectangular block whose boundary triangulation can be represented by 12 triangular patches.
2. Compute the swept volume of the cutting tool using the SDE method and generate the candidate points (ingress, grazing and egress) that represent the swept volume: the implementation of the SDE has been described in Chapter 4.
3. Construct the grazing curves of the swept volume. The grazing points are connected with lines.
4. Find the intersection points of the grazing curves with the virtual workpiece. The lines between grazing points in step 3 are used for finding the intersection points by checking if those lines intersect the virtual workpiece or not.
5. If the grazing curves intersect the workpiece, determine the grazing points interior to the workpiece by checking the intersection points location with respect to the grazing points location on the line (section 6.4.2). Create a list of points inside the workpiece from the intersection points and the grazing points.
6. Create triangular patches from the points computed in the step 5 by using the Delaunay algorithm.

7. Find the workpiece points outside the swept volume: a list of these the points can be obtained by subtracting the intersection points from the total workpiece points.
8. Create triangular patches from the points in step 6 by using Delaunay algorithm.
9. Combine the two triangulated surfaces from step 6 and step 8 by using the intersection lines as indicators of boundary connections.

Next, we describe the Delanuay algorithm for generating triangular patches from points and then the ray-casting method for determining when boundary points of one object are in the interior, the exterior or on the boundary of another object.

6.4.1 The Delanuay Algorithm

Although the original inventor of the Delaunay algorithm is not known, it is re-invented frequently because it provides a fundamental order for a set of multivariate data [Roger, 1964; Watson, 1985; Lawson, 1986 and Barber, 1996]. "Delaunay" is the name of a Russian mathematician, who used this method in the early 19 century to illustrate his ideas about the solar system [Watson, 1992].

For clarity, illustrations are confined to two dimensions, but these concepts are not limited to any particular dimension or to Euclidean space. No theorems or proofs will be presented because these are available in the literature. For any set of points $\{P_i, i = 0, \dots, N-1\}$ in n -dimensional space, $N > n$, there is a set of balls such that every portion of the convex hull of $\{P_i\}$ except the $\{P_i\}$ is contained within one or more balls. This often is referred to as the empty circumcircle criterion in the literature and the balls are sometimes called natural neighbor circles. A Delaunay complex is an aggregate of contiguous simplices whose vertices are the $\{P_i\}$ and whose circumspheres satisfy the

empty circumcircle criterion. A Delaunay complex is unique if the $\{P_{ij}\}$ are in general position.

A Delaunay complex has a geometrical dual- the set of Voronoi polytopes $\{\pi_{ij}\}$ associated with the $\{P_{ij}\}$. Each $\pi_i = \{x \in R^n: |x - P_i| < |x - P_j|, j=0, \dots, N-1, j \neq i\}$. A Voronoi tessellation is always unique, and unbounded because the $\{\pi_{ij}\}$ on the perimeter are unbounded. Each Voronoi polytope is an intersection of half spaces defined by perpendicular bisection hyper-planes between pairs of $\{P_{ij}\}$.

Many algorithms for constructing the Delaunay complex have been developed, and a useful approach involves the incremental insertion of the data into an existing complex. A datum to be inserted into the interior of an existing complex will be always

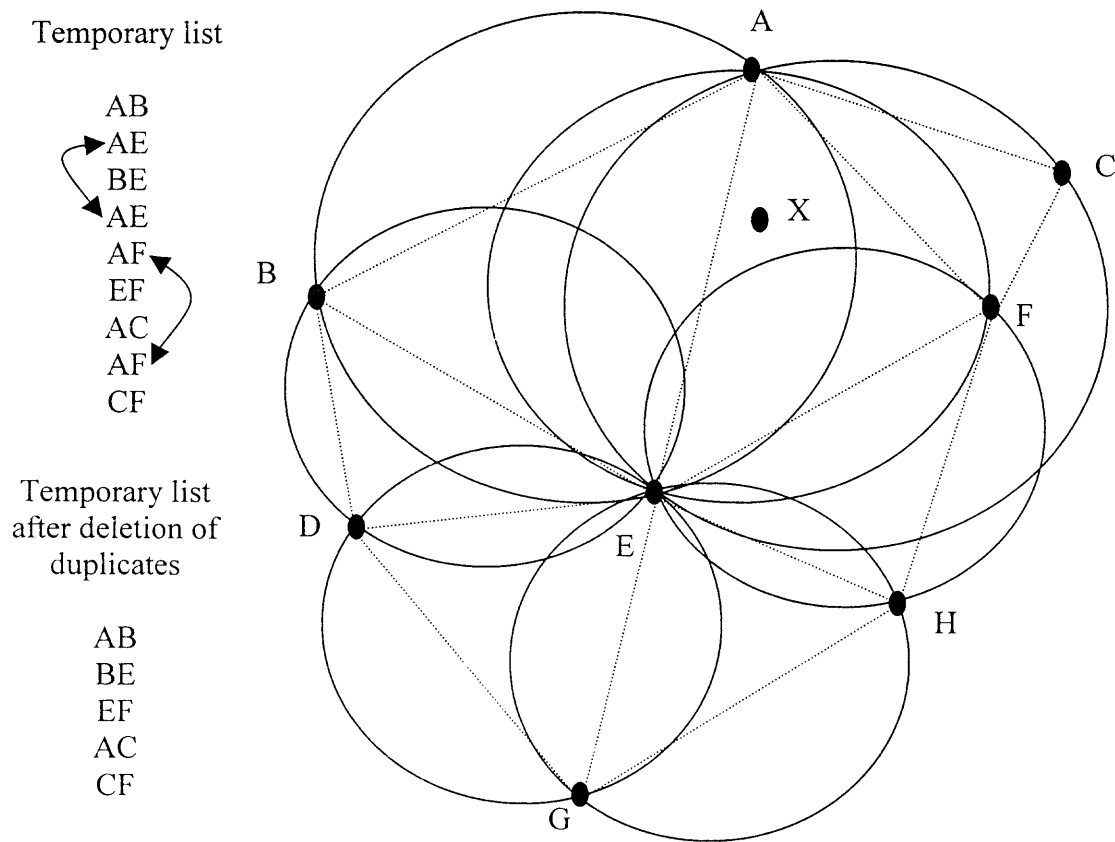


Figure 6.4 Implementation of the Delaunay algorithm

within one or more circumcircles as shown in Figure 6.4. The datum X is within the circumcircles of ABE, AEF and ACF. The procedure places the edges of these triangles on a temporary list; then the list is scanned and duplicate edges are deleted.

The triangles ABE, AEF and ACF with intersecting circumcircles are deleted and new triangles ABX, BEX, EFX, CFX and ACX are formed with the datum X and each of the edges remaining on the temporary list, as shown in Figure 6.5. This simple approach updates an existing Delaunay complex by finding and deleting all existing simplices that are affected by the addition of a new point, and then generating a replacement set of simplices involving the new point. The empty circumcircle criterion is always satisfied by insertions made in the manner.

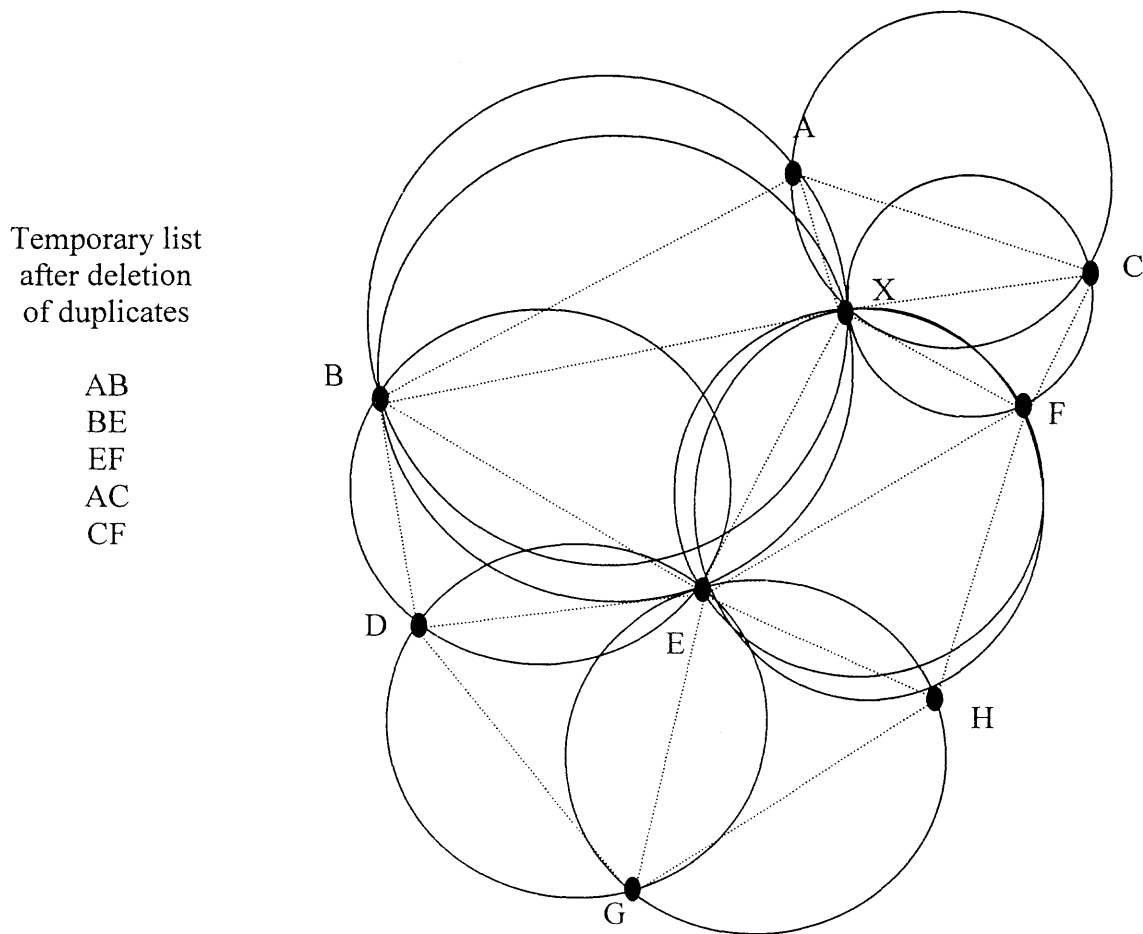


Figure 6.5 The new triangulation after inserting the datum X

6.4.2 Classifying the Points of the Object

The intersection of polyhedra may be classified by the ray-casting technique discussed in Chapter 4. We construct a vector in the direction of the surface normal of a polyhedron from a point in the interior, and then find the closest polygonal face of the boundary of the other object that intersects this vector. If no polygon is intersected, the original polyhedron is outside the other objects. If the closest intersecting polygon is coplanar with the original polyhedron, then this is a boundary-boundary intersection, and comparing the normals of polygon face determines the type of intersection. Otherwise, the dot product of the two polygonal face normals is computed. A positive dot product indicates that original polygon is inside the other object, whereas a negative dot product indicates that it is outside. A zero dot product occurs if the vector is in the plane of the intersection polygon; in this case, the vector is perturbed slightly and is intersected again with the other object's polygons.

Vertex-adjacency information can be used to avoid the overhead of classifying each polygon and points in this way. If a polygon is adjacent to (shares vertices with) a classified polygon and does not meet the surface of the other object, then it is assigned the same classification. All vertices on a common boundary between objects can be marked during the initial intersection line calculation.

Figure 6.6 shows an approach for finding the intersection points between the grazing points curve and the virtual workpiece and classifying if the grazing points are inside or outside the virtual workpiece. First step is to find the intersection points (A and B) and this can be done by simply finding the intersection points between the triangles

(plane) of the workpiece and the lines that connect grazing points. The intersection computation details have been described in section 4.4.2.

The locations of the intersection points determine the grazing points that are inside the object or outside the object. Grazing points (1&2) are outside the object because the locations of the intersection points (A&B) are out of the range of points (1&2). The points (5&6) are the same case of points (1&2). Grazing points (3 &4) are inside the object because their locations are between the intersection points.

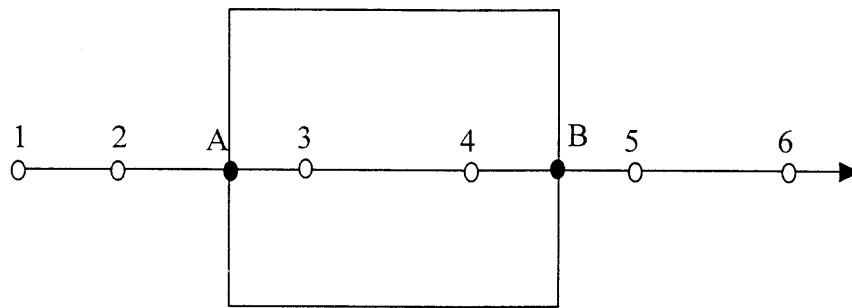


Figure 6.6 Classifying the grazing points

6.5 The Complexity of the Integration Methods

The efficiency of the insertion Delaunay algorithm described above, as expressed by the order of complexity depends upon the number of circumspheres that must be tested for containment. Let m be the number of input points in R^n and r the number of processed points, then the worst case complexity of the algorithm is $O(m \log r)$. If $r = m$ then the cost of the method is $O(m \log m)$ [Barber, et al., 1996].

The computation complexity (or cost) of the SDE method for generating the ingress, grazing and egress points is order of $O(N^{1/3})$ without triangulation of the points. Here $N^{1/3}$ is the number of the vertices of the swept object after triangulation and also an

upper bound for the number of times steps. Triangulation of the points (ingress, grazing and egress) using the Delaunay method costs $O(N \log N)$. So the total computation cost of the SDE method with triangulation is order of $O(N^{7/3} \log N)$. But the triangulation method described in Chapter 4, in which the number of grazing points is kept fixed, reduces the SDE complexity to order $O(N^{4/3})$.

The computational cost of the classification method is linearly proportional to the number of the polygons in both solids, so it is $O(k)$, where k is the number of polygons in both solids. Hence the cost is $O(N)$.

The total complexity of integrating of the SDE method with the interactive Boolean operation method can be calculated by following the steps of the integration in Figure 6.3. The computational complexity of the integration method algorithm, which includes computation of the swept volume and the Boolean operations is less than or equal $O(N^{10/3} \log N)$. Thus the swept volume and Boolean operations can be both be performed at a cost of no more than $O(N)$ more than computation of the swept volume alone.

CHAPTER 7

CONCLUSIONS AND FUTURE WORK

In this research, we have obtained a large body of results. Beyond these immediate results, however, our research has also produced some more fundamental improvements in our understanding of the design and surface creation model by using the VR environment. In this chapter, we will briefly discuss several of these contributions. Also we discuss future work in this field.

7.1 Conclusions and Major Contributions

A novel approach has been introduced for creating freeform surfaces and objects. This approach creates a Virtual Reality environment for the user or the designer to manipulate virtual tools to carve a design model in a 3D space. Several algorithms have been developed and integrated in order to meet the VR requirement and give users the flexibility to convert their ideas into computer representations.

A prototype system has been developed for the VR environment for the creation of freeform surfaces and objects. The prototype system has been used to study new ways of creating design models. VR hardware (3D motion tracker device, audio device and display device) and a user interface have been developed and integrated with the VR system to assist in the communication of the user's thoughts into computer representation.

A solid modeling engine has been developed to support the VR requirements in terms of real-time and flexibility. The solid modeling engine is based on two main components, which are the SDE method to compute the swept volumes of moving objects

(virtual tools) in space and the ray-casting method to perform the Boolean subtraction between the swept volumes and virtual workpiece. A new algorithm for Boolean subtractions that is specifically adapted to the use of the SDE method has been developed. Three techniques (improved bounding box, scan-rendering and data mechanism) have been developed and integrated with the solid modeling engine to speed up the computation time. An improved bounding box method has been developed and adapted to the SDE approach to reduce the number of pixels for computation and data manipulation. Scan-rendering techniques have been combined with the SDE method to reduce intersection calculations. A new data mechanism has been developed to reduce the computation cost of transformation and data exchange between the SDE method and ray-casting.

A method has been developed to convert the pixel representation into NURBS representation in order to integrate the VR system for creation of freeform surfaces with existing commercial CAD/CAM systems. The method converts the virtual design model from pixel representation into contour curves. These curves are then converted to B-spline curves and input to Pro/Engineer (commercial CAD/CAM package).

An approach for selection of geometric resolution parameters in the virtual design model creation, which uses flat-end and ball-nose virtual tools with discrete representation of the workpiece and cutting edge geometry, is presented. An empirical measurement is performed to quantify how the positional uncertainty of the motion tracker in the VR system affects the representation accuracy of the design model. An algorithm has been developed to control the accuracy of the design model. Illustrative examples are presented to demonstrate how the VR system actually works.

An Interactive Boolean Operations algorithm for designing 3D Solids in a Virtual Reality environment has been developed. This method can be used in our VR approach to create virtual design models with an accuracy equal to the approximation accuracy of the SDE method at a computational cost (time) roughly the same as that of the standard SDE algorithm. The method has the following ordered steps: 1) Classify the vertices of one object relative to the other to determine whether they lie inside, outside or on the boundary and then add the intersection vertices to obtain a new set of vertices 2) Construct an efficient triangulation of the intersected object using the new set of vertices. The computational complexity of the integration method algorithm, which includes computation of the swept volume and the Boolean operations is $O(N^{10/3} \log N)$. Thus the swept volume and Boolean operations can be both performed at a cost of no more than $O(N)$ more than computation of the swept volume alone.

7.2 Future Work

The VR approach for creation of freeform surface and objects can serve as the next generation of CAD/CAM system by expanding research work in the following areas:

- 1- Add a haptic device to our VR system and study the effect on the design model process.
- 2- Add more virtual tools, which simulate other manufacturing processes.
- 3- Study more deeply the process of generating the NC program directly from the design process.

REFERENCES

1. Abdel-Malek, K. and Yeh, H. J., Geometric representation of the swept volume using Jacobian rank-deficiency conditions, *Computer-Aided Design*, 29(6): 457-468, 1997.
2. Angster, S., Jayaram S., VEDAM: Virtual environments for design and manufacturing, *ASME Symposium on Virtual Reality in Manufacturing Research and Education*, 21-31, Oct. 7-8, 1996.
3. Ascension Technology Corporation, Installation and operation guide, Ascension Technology corporation, Burlington, Vermont, 1999.
4. Atherton, P., Earl, C. and Fred, C., A graphical simulation system for dynamic five-axis NC verification, *Proceedings Autofact'87*, 2-1-2-12, 1987.
5. Barber, B. C., Dobkin, D. P. and Huhnpaa, H., The quickhull algorithm for convex hulls, *ACM Transactions on Mathematical Software*, 22: 469-483, 1996.
6. Barrus, J.W., The Virtual Workshop: A Simulated Environment for Mechanical Design, PhD Thesis, Department of Mechanical Engineering, Massachusetts Institute of Technology, September 1993.
7. Barrus, J. W. and Woodie F., A simulated environment for mechanical design, *ASME Symposium on Virtual Reality in Manufacturing Research and Education*, 89-99, Oct. 7-8, 1996.
8. Bayliss, G., Bowyer, A., Taylor, R., and Willis, P., A virtual factory, *International Journal of Shape Modelling*, 110-120, 1997.
9. Blackmore, D., and Leu, M.C., Analysis of swept volume via Lie groups and differential equations, *International Journal of Robotics Research*, 11: 516-537, 1992.

REFERENCES

(Continued)

14. Bowman, D., Wineman, J., Hodges, L., and Allison, D. Designing animal habitats within an immersive VE. IEEE Computer Graphics and Applications, vol. 18, no. 5, pp. 9-13, September/October 1998.
15. Christensen, D., Sculpting Virtual Reality, Science News, 156(12):177-192, September 18, 1999.
16. Dani, T. H., Chu, C.P., and Gadh, R., COVIRDS: shape modeling in a virtual reality environment, ASME Design Engineering Technical Conference, Sacramento California, September 14-17, DETC09/CIE4302, 1997.
17. Deng, Z., Leu, M.C., and Blackmore, D., Application of sweep differential equation approach to 5-axis sculptured surface machining, Proceedings of the Third International Conference on Automation Technology, Taipei, Taiwan, 1994.
18. Deng, Z., Leu, M.C., Wang, L. and Blackmore, D., Determination of flat-end cutter orientation in 5-axis machining, Proceedings of Manufacturing Science and Engineering-1996, MED-Vol. 4, pp. 73-80, 1996 ASME International Mechanical Engineering Congress and Exposition, Atlanta, Georgia, Nov.17-22, 1996.
19. Deviprasad, T. and Kesavadas, T., VPAVE: an interactive tool for validating assembly components in virtual environment using finite element simulation, ASME Symposium on Virtual Environment for Manufacturing, 73-83, 1999.
20. Fang, X. D., Luo, S., Lee, N. J., Jin, F., Virtual Machining Lab for Knowledge Learning and Skills Training, John Wiley & Sons, 89-97, 1998.
21. Foley, J.D., Van Dam, A., Feiner, S. K., Hughes, J. F., Computer Graphics: Principles and Practice, Addison-Wesley Publishing company, Inc., 1997.
22. Furlong, T. J., Virtual reality sculpture using freeform surface deformation, ASME Design Engineering Technical Conference, Sacramento California, September 14-17, DETC09/CIE4302, 1997.
23. Goldstein, R. A. and Nagel, R., 3-D visual simulation, Journal of Simulation, 16(1): 25-31, Jan. 1997.
24. Gossard, D.C., Analogic Part Programming with Interactive Graphics, PhD thesis, Department of Mechanical Engineering, Massachusetts Institute of Technology, February 1975.

REFERENCES (Continued)

25. Hook, T. Van., Real-time shaded NC milling display, In SIGGRAPH '86, 20(4): 15-20, 1986.
26. Hsu, P.L. and Yang, W.T., Realtime 3D simulation of 3-axis milling using isometric projection, Computer-Aided Design, 25(4):215--224, April 1993.
27. Jiang, H., The flow approach to swept volume, Masters Thesis, Department of Mathematical Sciences, New Jersey Institute of Technology, Newark, New Jersey, 1993.
28. Kameyama, K., Virtual clay modeling system, ACM VRST'97, 197-200, Lausanne, Switzerland, 1997.
29. Keppel, E., Approximating complex surfaces by triangulation of contour lines, IBM Journal of Research and Development, 2-11, 1975.
30. Kimura, F., Virtual manufacturing as a basis for concurrent engineering, IFIP Transactions (B) Applications in Technology, 17:103-117, 1994.
31. Laidlaw, D.H., Turumbore, W.B. and Hughes, J.F., Constructive solid geometry for polyhedral objects, SIGGRAPH 86, 161-170, 1986.
32. Lawson, C. L., Properties of n-dimensional triangulations, Computer-Aided Geometric Design, 3: 231-246, 1986.
33. Leu, M. C., Blackmore, D., and Maiteh, B., Deformed swept volume analysis of NC machining simulation with cutter deflection, Proceedings of Sculptured Surface Machining Conference, Auburn Hills, MI, Nov. 9-11, 1998.
34. Leu, M.C., Park, S.H., and Wang, K.K., Geometric representation of translational swept volumes and its applications, Journal of Engineering for Industry, 108(2):113-119, May 1986.
35. Leu, M.C., Blackmore, D., Wang, L. and Pak, K., Implementation of SDE method to represent cutter swept volumes in 5-axis NC milling, Proceedings of the International Conference on Intelligent Manufacturing, 211-220, Wuhan, China, June 14-18, 1995.
36. Leu, M.C., Wang, L., and Blackmore, D., A verification program for 5-axis NC machining with general APT tools, Annals of CIRP, 46: 419-424, 1997.
37. McLean, C.R., Bloom, H.M., and Hopp, T.H., The virtual manufacturing cell, Proceeding of the 4th IFAC/IFIP Symposium in Information Control Problems in Manufacturing Technology, pages 207-215. IFAC, October 1983.

REFERENCES

(Continued)

38. Menon, J.P. and Robinson, D.M., Advanced NC verification via massively parallel raycasting, *Manufacturing Review*, vol. 6, no. 2, pp.141-154, 1993.
39. Miller, J.P, Geometric approaches to non-planar quadric surfaces intersection curves, *ACM TOG*, 6(4): 274-307, October 1987.
40. Mills, J.J., Graham, J.K., Elmasri, R.A., and Weems, B.P., The virtual manufacturing workbench: Representation and interface issues, *IFIP Transactions (B) Applications in Technology*, 10:231-244, 1993.
41. Narvekar, A.P., Representation and application of swept solids for numerically controlled milling, Masters Thesis, Department of Mechanical and Aerospace Engineering, State University of New York at Buffalo, Buffalo, New York, 1991.
42. Neider, J., Davis, T. and Woo, M., *OpenGL Programming Guide*, Addison-Wesley Publishing Company, 1993.
43. Oliver, J.H. and Goodman, E.D., Graphical verification on NC milling programs for sculptured surface parts, *Symposium on Integrated and Intelligent Manufacturing*, pages 247-263. ASME, 1986.
44. Oliver J.H. and Goodman E.D., Direct dimensional NC verification, *Computer-Aided Design*, 22(1):3-9, Jan/Feb 1990.
45. Perles, B.P. and Vance, J. M., Interactive virtual tools for manipulating NURBS surfaces in a virtual environment, *ASME Symposium on Virtual Environment for Manufacturing*, 85-91, 1999.
46. Qin, D., Blackmore, D., and Leu, M.C., "Improved flow approach for swept volumes", *Proceedings of Japan-U.S.A. Symposium on Flexible Automation*, 1994.
47. Requicha, A.A.G, Mathematical models of rigid solids, Technical memo 28, Production Automation Project, University of Rochester, NY, 1997.
48. Rogers, C.A., *Packing and Covering*, Cambridge Tracts in Mathematics and Mathematical Physics, 54, Cambridge University Press, 1964.
49. Roth, S.D., Stereo 3-D perception for a robot, Ph.D. thesis, California Institute of Technology, Pasadena, California, March 1978.
50. Saito, T. and Takahashi, T., NC Machining with G-buffer method, *Computer Graphics*, 25(4): 207-216, 1991.

REFERENCES

(Continued)

51. Sambandan, K., Graphic simulation and verification of five-axis NC machining, M.S. Thesis, School of Mechanical and Aerospace Engineering, Cornell University, Ithaca, New York, 1988.
52. Sambandan, K., Geometry generated by sweeps of polygons and polyhedra, Ph.D. Dissertation, School of Mechanical and Aerospace Engineering, Cornell University, Ithaca, New York, 1990.
53. Smith, S. and Tlustry, J., An overview of modeling and simulation of the milling process, ASME Journal of Engineering for Industry, 113:169-175, 1991.
54. Smithers, T., AI-based design versus geometry-based design, or why design cannot be supported by geometry alone, Computer-Aided Design, 21(3): 141-150, 1989.
55. Sarraga, R.F., Algebraic methods for intersections of quadric surfaces in GMSOLID, CVGIP, 22(2): 222-238, 1983.
56. Sungurtekin, U.A. and Voelcker, H.B., Graphical simulation and automatic verification of NC machining programs, Proceedings IEEE Robotics & Automation Conference, 156-165, April 1986.
57. Takata, S., Tsai, M.D., Sata, T., and Inui, M., A cutting simulation system for machinability evaluation using a workpiece model, Annals of CIRP, 38: 417-420, 1989.
58. Taylor, R., Bayliss, G., Bowyer, A., Willis, P., Virtual workshop for design by manufacture., ASME Database Symposium, 921-926, Sep. 17-20, 1995.
59. Toriya, H., Chiyokura, H., Foundation and application of 3D CAD, Kyouritsu Shuppan, Tokyo, 1991.
60. Wallis, A.F., Tool path Verification using Set-Theoretic Solid Modeling, Ph.D. Thesis, School of Mechanical Engineering, University of Bath, 1991.
61. Wang, W.P. and Wang, K.K., Geometric modeling for swept volumes of moving solids, IEEE Computer Graphics & Applications, 6(12): 8-17, December 1986a.
62. Wang, W. P. and Wang, K.K., Real time verification of multi-axis NC programs with raster graphics, Proceedings IEEE Robotics & Automation Conference, pages 166-171, April 1986b.
63. Watson, D.F., Natural neighbor sorting, The Australian Computer Journal, 17(4): 189-193, 1985.

REFERENCES

(Continued)

64. Watson, D.F., Contouring: a guide to the analysis and display of spatial data, Pergamon press, 1992.
65. Weld, J. D., and Leu, M. C., Geometric representation of swept volumes with applications to polyhedral objects, International Journal Robotics , vol. 9, pp.105-117, 1990.
66. Zeid, I., CAD/CAM theory and practice, McGraw-Hill, Inc.,1991.
67. Zetu, D., Banerjee, P. and Akgunduz, A., Telemetry-based depth recovery for virtual factory construction and extension to remote facility management, ASME Symposium on Virtual Environment for Manufacturing, 73-83, 1999.