Abstract

ZHU, WEIHANG, Virtual Sculpting and Polyhedral Machining Planning System with Haptic Interface (Under the direction of Dr. Yuan-Shin Lee)

This research proposes the methodology of a novel haptic sculpting and machining planning system for virtual prototyping and manufacturing. A lab-built 6-DOF (degree of freedom) input and 5-DOF output haptic interface system is utilized in the proposed haptic sculpting and machining planning system. A dexel-based haptic virtual prototyping CAD system and a triangulated surface-based machining planning (manufacturing) system are developed. A *dexel-based collision detection method* and a *force-torque feedback analysis* are proposed for virtual prototyping module. The output of the virtual prototyping module can be either STL polyhedral surface model, or the tool motion, which is recorded as NC (numerically-controlled) commands. Haptic interface is also used in the machining planning to help determine the feasible tool orientation for 5-axis NC tool path generation. A new machining strategy of 5-axis *pencil-cut* machining is proposed with the haptic interface. An OBB (object bounding box)-tree and point-cloud-based *Two-phase collision detection* and *force-torque feedback algorithm* are proposed for virtual manufacturing module. *Dexel-based method* is developed for global tool interference avoidance with other components in a complex machining environment. To bridge the virtual prototyping module and machining planning module, a *conversion marching algorithm* is proposed to construct STL surface models from Dexel volume models. The algorithm can be used in both virtual prototyping system and NC simulation and verification. In the virtual sculpting module, a user can virtually sculpt a stock volume material intuitively by the haptic interface system. Hardware and software implementations of the haptic sculpting and machining planning system are also presented in this paper. The proposed methodology and developed haptic sculpting and machining planning system can be used in CAD/CAM systems and virtual prototyping.

Virtual Sculpting and Polyhedral Machining Planning System with Haptic Interface

By

Weihang Zhu

A dissertation submitted to the Graduate Faculty of

North Carolina State University

In partial fulfillment of the

Requirements for the Degree of

Doctor of Philosophy

Industrial Engineering

Raleigh

Summer, 2003

Approved by:

_____	_____
Dr. Yuan-Shin Lee	Dr. Ezat T. Sanii
Chair of Advisory Committee	Advisory Committee
_____	_____
Dr. Robert E. Young	Dr. Christopher G. Healey
Advisory Committee	Advisory Committee

Biography

Weihang Zhu was born in May, 1976 in Hangzhou, China. He received his Bachelor degree from Department of Energy Engineering in 1997 and Master Degree from College of Mechanical and Energy Engineering in 2000, all from Zhejiang University, Hangzhou, China. He was a network engineer of China Telecom for months before he started his Ph.D graduate study in the Department of Industrial Engineering at North Carolina State University, USA. His research interests include computer haptics, haptic applications in CAD/CAM, multi-axis sculptured surface machining, polyhedral model machining, applied computational geometry, discrete-event computer simulation, and logistics.

Acknowledgements

My memory overflows when it comes to acknowledgement. There are too many people I am indebted to in this world. First I would like to thank my family, especially my parents, for their sustained moral and material support throughout my life. A floating leaf in this end of the planet, I have always been feeling their support and care from the other end of the Earth. Thank you, Dad and Mom!

I would not have made today's achievement, even though very little in retrospect, without the guidance and support from my advisor and friend, Dr. Yuan-Shin Lee. He has guided me through my transition from an Energy Engineer to an Industrial Engineering researcher. His continuous encouragement has inspired me to cross the Rubicon in different stages of my research. Besides his great support in my study and research, I have benefited a lot from his life philosophy, which is to accompany me in the rest of my life.

I have been working as a Teaching Assistant for Dr. E. T. Sanii, who is serving in my committee. Not only did I learn a lot about teaching from him, but I was very happy to make a sincere friend as Dr. Sanii as well. Dr. R. E. Young has always been the role model of kindness in my mind and I was glad that I got enrolled and excelled in his class in my first semester in USA. Dr. C. G. Healey has been an outstanding researcher even since and I would like to thank him for his help not only in serving in my committee but in providing a lot of technical support as well. I should also thank Dr. Y. Fathi to be the substitute in my Preliminary exam (summer 2002) by squeezing some precious time from his busy life. I would like to thank Dr. D. Kaber for his support in the haptic controller, without which I could not proceed with my research. I would like to thank Dr M. Y. Chow to serve as the substitute of Graduate School Representative (summer 2002). I am heavily indebted to Dr. Y. Adachi from *Suzuki Inc.*, Japan. His patient instructions guided me through the door of haptic research.

I would like to thank my friend, Dr. Yongfu Ren, the number one programmer and designer in IE department, for his invaluable discussion and suggestion. Without him it was impossible for me to finish my dissertation or papers in a short time. For sure I would like to thank the other group mates of all ages: Dr. Jirawan Kloypayan, Susana Lai-Yuen, Dr. Bahattin Koc, Abhinand Pamali, Ron Aman, Chirag Chafekar, Jennifer Hartman. I am proud that I can write down all the names correctly. ☺ From Portland Conference, I am also glad to know Dr. C. J. Chiou, an alumnus of our research group, who is very kind and also helped me a lot. I would like to thank Professor C. S. Jun, Professor H. T. Yau and Professor S. Y. Ju for their generous help during their sabbatical leave in our lab.

Mr. Jason Low, our manufacturing lab technician, is probably the most versatile people in our Parkshops Lab. I would like to thank him for his great support during my own class and my TA class. Mr. Darrell Rice, our manufacturing lab manager, is also very helpful through my lab assignments.

I have a whole bunch of good friends and classmates to thank, both in USA and China. I have enjoyed their friendship and I believe that they have enjoyed mine, too. Don't get me wrong that I am an ingrate by not listing all their names, but my memory overflows. However, they are all in my heart. Thank you all!

Table of Contents

List of Figures

Figure 3.10 Tool cutting profile of different kinds of tools 38

Figure 3.11 Tool move up and down along the tool axis 41

Figure 3.12 Tool rotates only, without translation 41

Figure 3.13 Tool moves linearly without rotation: tool swept volume breakdown 41

Figure 3.10 Tool cutting profile of different kinds of tools 38

Figure 3.11 Tool move up and down along the tool axis 41

Figure 3.12 Tool rotates only, without translation 41

Figure 3.13 Tool moves linearly without rotation: tool swept volume breakdown 41

List of Tables

Chapter 1
Introduction

1.1 Motivation and research objectives

In reality artists create the intuitive art works to make the world more wonderful. Every so often CAD (Computer-aided Design) system users wish to have an intuitive way to create something that a normal CAD system cannot create easily. This research focuses on helping them to pragmatize this dream in the virtual world, not only for the intuitive design, but also for the manufacturing of the designed models. A haptic sculpting and machining planning system for virtual prototyping and manufacturing has been developed following this idea. In the system, a user can virtually sculpt a stock to get an intuitive design by using a haptic interface. The designed model is then processed by the machining planning (manufacturing) module of the system. Haptic interface is also involved in the machining planning to help determine the tool orientation in 5-axis NC (numerically-controlled) tool path generation.

The word '*haptic*' means 'relating to or based on the sense of touch'. Basically haptic interface provides force-feedback in a Virtual Environment. In this research, the haptic system provides force and torque feedback to the user during the virtual sculpting and machining planning processes, thereby facilitating the execution of the tasks.

The haptics research *per se* can be generally classified into three parts: *human haptics*, *haptic interface design* and *Computer Haptics* (haptic rendering) [Srinivasan 1997].

- ✧ *Human haptics* deals with the ergonomic issues of the haptics research. It studies the mechanical, sensory, motor and cognitive components of the hand-brain system. Although this is an important part of haptics research, it is out of the scope of this research.
- ✧ With the rapid development in the computer and electronics industry, the design and manufacturing of *haptic interface hardware* has improved

dramatically. Commercial haptic hardware has become available. The haptic devices can be used in different fields as medical, computing, entertainment, automotive, 3D interaction, and so on. The research based on the R&D (research and development) of University of Utah provides advanced robotic systems, medical devices, and Mechanical and Electrical Microsystems for practical applications. Besides the commercial products, some R&D centers of big companies have also been making their own haptic interface and doing research on both its manufacturing and practical applications. For example, the Virtual Reality group in *Suzuki Inc.* (Japan), led by Dr Y. Adachi, is customizing various types of haptic interfaces for the research and applications. The haptic device used in this research is from *Suzuki Inc*.

- ✧ *Computer Haptics* is concerned with generating and rendering haptic stimuli to the human user. It is also commonly referred as Haptic Rendering, as it is analogous to the Graphics rendering in Computer Graphics. Compared to Graphics rendering, Haptic Rendering requires much higher update rate (1000Hz) in order to provide a constant feeling of force feedback. Generally this topic spans *Object modeling* and *Collision Detection and Response*. The stability of haptic interface response is also an important issue. It is the combined result of haptic hardware, Haptic Rendering software and human response.

The idea of *Virtual Sculpting* has been there for more than a decade. Earlier work at least dates back to 1991, when Galyean and Hughes [Gaylean 1991] presented their interactive modeling technique based on the notion of sculpting a solid material. A primitive 'poor man' force feedback unit was utilized as part of the interface for the system. The idea flourished in many consequential papers as in [Dachille 2001, Leu 2002, and Wang 1995], to name a few. This kind of system attempts to complement the commercially available CAD (Computer-aided Design) system. It allows the users to implement their design ideas in an intuitive and easy manner with an advanced 3D force feedback interface.

The idea of *Machining Planning* with a haptic interface is relatively new. Some initial attempts and inspiring work have been done by Sarma's research group at MIT [Ho 2001]. They have produced some interesting results in 5-axis tool path generation. In their work, a quick collision detection method was proposed between a tool and an environment represented by point clouds. In their 5-axis tool path generation application, they machined a part with constant Z-height machining method [Balasubra 2002]. Five-axis NC (numerical controlled) machining is widely used to produce complex surfaced parts, such as turbine blades, aerospace parts, dies and molds. With two additional degrees of freedom than the traditional 3-axis machining, 5-axis machining offers many advantages over 3-axis, including better tool accessibility, faster material removal rates and improved surface finish [Choi 98]. However, to make best use of 5-axis machining, we need to resolve more complicated interference (gouging and collision) problems and also determine the optimal tool orientations for complex surface machining [Jun 2002]. As a novel idea, haptic interface is proposed to help in determining the global optimal tool orientation in this research.

With the rapid development of haptic research, more exciting possibilities open up to various applications of virtual reality and teleoperation. Haptic interface has been introduced into various fields such as medicine, entertainment, education, industry, graphic art and so on. It is also finding its way into the arena of CAD/CAM, which is the main objective of this research. It is expected that the current research will instigate more interesting haptic research and application in the CAD/CAM area in the next decades.

1.2 Dissertation Organization

In this dissertation, the following research issues are addressed:

- *Analytical Methodology of Dexel-based Virtual Sculpting with a 5-DOF Haptic Interface*: There have been several methodologies for virtual sculpting in the past decade. This research proposes an analytical methodology for virtual sculpting with a 5-DOF Haptic Interface. The idea stems from the NC (Numerically-Controlled) simulation and verification process. A virtual sculpting system at least consists of a tool and a stock material. The stock

material is represented with Dexels (small building blocks). Swept volume of a virtual tool is used to update the Dexels of the stock material. By doing so, not only the shape of the stock volume material is changed, but the force and torque response is calculated and fed back to a human user. An object collision detection method and a force-torque feedback algorithm are proposed according to the tool motion analysis.

- *Machining Planning System with Haptic Interface*: Machining planning system is in the area of CAM (Computer-aided Manufacturing). Models designed in CAD system or virtual sculpting system can be imported into the machining planning system. NC (Numerically-Controlled) code can be generated from the machining planning to realize the CAD design in manufacturing. It is used to help determine the tool orientation in 5-axis NC tool path generation. A special machining strategy, *5-axis Pencil-cut*, is proposed with haptic interface. The result of the effort shows the promising utilization of haptic interface in the CAD/CAM field.
- *Bridging Virtual Sculpting and Machining Planning System -- Constructing polyhedral models from Dexel volume models for haptic virtual sculpting*: STL polyhedral surface model is used in machining planning system. Dexel volume model is used as the in-process model during the virtual sculpting. A *conversion marching algorithm* is proposed to construct STL surface models from Dexel volume models, so that they can be processed by available CAM (Computer-aided Manufacturing) or RP (Rapid Prototyping) systems.

The remaining of the thesis proposal is organized as follows:

Chapter 2 presents literature review and research background on several related issues, including the haptic interface, virtual sculpting and polyhedral model machining planning.

Chapter 3 proposes an analytical methodology for virtual sculpting with a 5-DOF (Degree of Freedom) output haptic interface.

Chapter 4 discusses the machining planning system with a haptic interface. A new machining strategy, *5-axis Pencil-cut* is proposed with the aid of a haptic interface.

Chapter 5 talks about the constructing STL surface models from Dexel volume models. This bridges the gap between virtual sculpting system and machining planning system.

Chapter 6 is the showcase of the developed virtual sculpting and machining planning system with haptic interface.

Chapter 7 concludes this research and suggests some future research.

Chapter 2

Literature Review and Research Background

In this chapter, the main issues involved in this research are discussed. They include haptic interface research, virtual sculpting and polyhedral model machining planning. These issues will be further discussed and referred frequently in the following chapters.

2.1 Haptics Research

The main focus of this research is on the utilization of haptic interface in the CAD/CAM application. For this reason, the haptic interface will be described in this section. Haptics is concerned with information and object manipulation through touch [Biggs 2002]. As mentioned in Chapter 1, the haptics research *per se* can be generally classified into 3 parts: *Human Haptics*, *Haptic Interface design* and *Computer Haptics* (*Haptic Rendering*) [Srinivasan 1997]. *Human haptics* deals with the ergonomic issues of the haptics research [Burdea 1996]. It is out of the scope of this research. *Haptic Interface design* is concerned with the haptic hardware [Burdea 1996]. Although the haptic interface involved in this research is designed and manufactured in *Suzuki Inc*. Japan, we have gone through the process of building its controller and developing its software driver. Hence the experience is also presented here, on purpose of helping the reader to understand the methodologies and the system to be presented in this dissertation. This research focuses on *Computer Haptics* and its applications in CAD/CAM.

2.1.1 Haptic interface hardware

The keyboard, mouse, and trackball are familiar, passive, non-haptic interfaces that sense a user's hand movements. Although they apply forces on the user's hand

and consequently provide tactual sensation, the forces are not under program control. Active haptic interfaces, such as desktop robots and exo-skeletal gloves with force feedback, are more sophisticated devices that have both sensors and actuators. In addition to transducing position and motion commands from the user, these devices can present controlled forces to the user, allowing him or her to feel virtual objects, as well as control them [Biggs 2002]. These kinds of device are what we usually refer to as the haptic device.

One classification of Haptic Interface hardware is whether it is ground-based or body-based [Srinivasan 1997], (non-portable or portable [Burdea 1996]). Ground-based force feedback devices are usually grounded somewhere like a desktop, a floor or a ceiling. This kind of devices includes Joysticks, Pen-based Masters, String Force-Feedback Interface, Robot arms, Floor-Grounded Exoskeletons *etc*. Body-based force feedback devices are usually smaller than ground-based devices. This kind of devices includes Arm Exoskeletons, Hand Master, *CyberForce* Glove *etc*. Hybrid devices combining both characteristics have also been built.

Another classification of Haptic Interface hardware is whether it is tactile display or net force display [Srinivasan 1997]. Tactile displays are the simulation of direct contact with virtual objects in Virtual Environments. Net force displays is relatively easier as the interaction is through a tool, e.g., feeling the virtual world with a rigid stick.

As shown in Figure 2.1, the Haptic Interface involved in this research is a 6-DOF input and 5-DOF output haptic device, made in *Suzuki Inc*. It is a pen-based net force display device and grounded to a desktop. Basically it has a left articulated arm and a right articulated arm. The two arms link to each other at the distal end, where a probe is located. The probe (grip) serves as the end-effector. The arms are driven by the 6 DC motors. These DC motors can provide force feedback on the X, Y, Z axes and torque feedback on the A, B axes. Hence it can provide 5-DOF output. The 6^{th} DOF is for the rotation of the probe itself and is meaningful for sensing. This DOF is not used in the current research, though, since 5-DOF is already enough for the current research. There are encoders attached to motors and also at the end of the

probe (grip). These encoders are used to provide position and orientation information of the probe. DC motors, encoders, articulated arms and probe (grip) are all attached to a strong desktop frame.

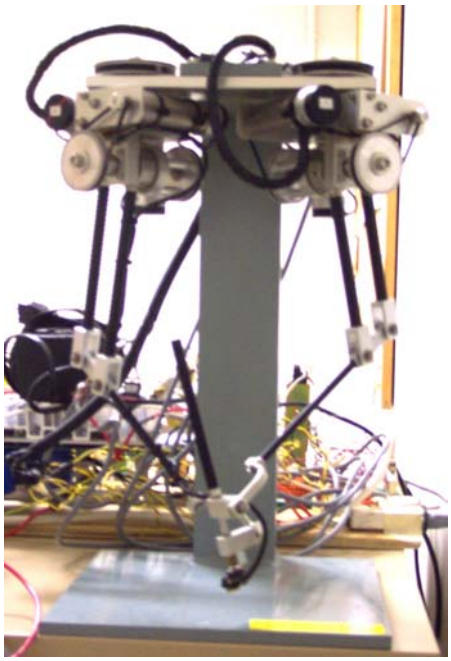


Figure 2.1. Our 5-DOF pen-based haptic device (Photo)

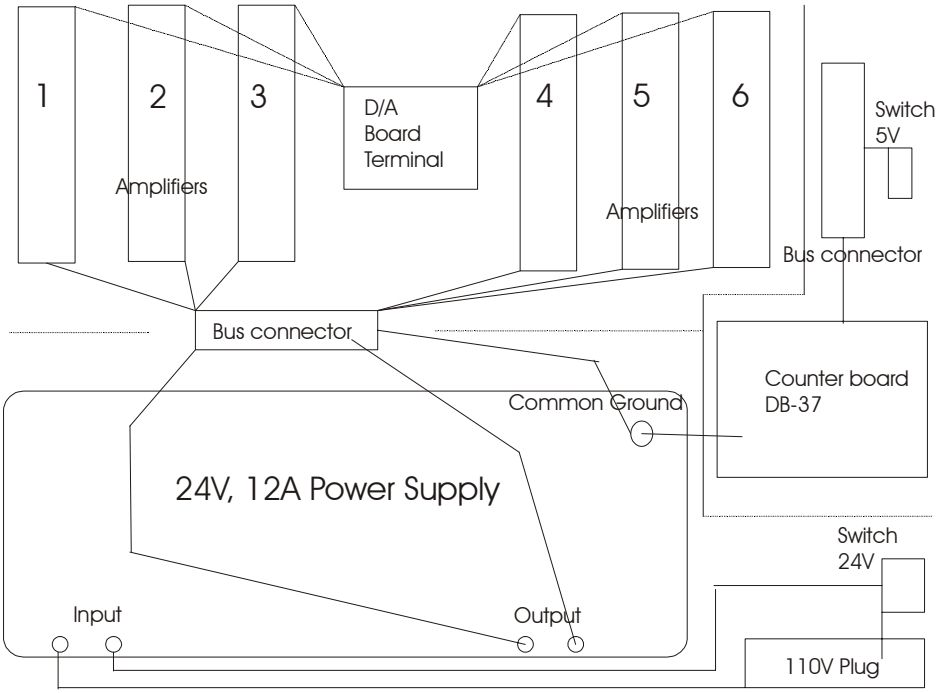


Figure 2.2. The haptic device controller layout

The above is the haptic device itself. What's behind it is the haptic controller. The commercial *PHANToM* device uses parallel port as the interface between the computer and the device. Our device is more powerful and can provide larger force than *PHANToM* device. Consequently, the haptic controller is much larger and complex. This haptic controller is made in our lab. Figure 2.2 shows the layout of the haptic controller. It can be generally divided into two divisions: input and output. The input division includes the counter board and its accessories. It accepts the signals from the encoders, which are attached to the motors. The output division includes the D/A (Digital / Analog) board, Amplifier, Power Supply and a few accessories. This division translates the computer commands into driving currents to drive the motors.

Figure 2.3 shows the relationship between the controller and the haptic device. A workstation with dual-CPU is used for running the CAD/CAM software and the control for the haptic device at the same time. Two boards, namely D/A (Digital / Analog conversion) board and counter board, are used for the communication between the computer and the haptic device.

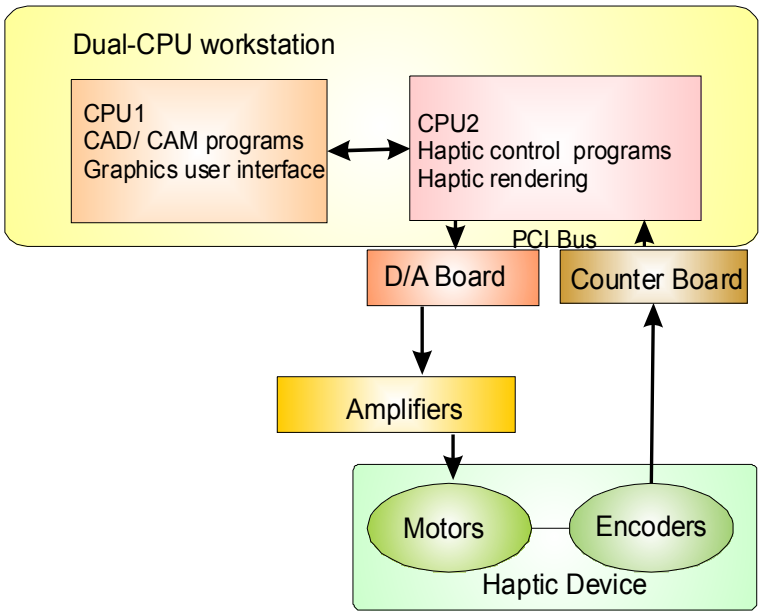
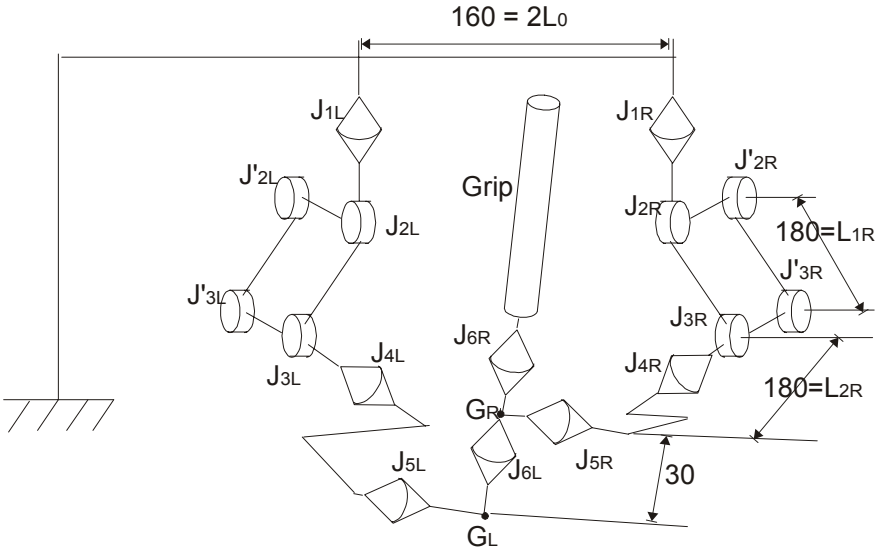


Figure 2.3. The haptic device hardware working loop

The working loop is described as follows. The user moves the probe and then the position and orientation information is picked up by the encoders. The encoders

send the information to the counter board, which counts the signal impulses. The CPU determines the position and orientation of the probe and thereby determines its relative posture in the virtual environment. If collision with other objects is detected, the depth of penetration and feed back force / torque are calculated by *Physically-based Modeling*. The required force and torque are transformed to equivalent motor torques by using *Jacobian Matrix* (see *Appendix A*). The required motor torques are transformed to the equivalent electronic signals, which are sent to the DC servo motor amplifiers via the D/A board. The 6 amplifiers, accordingly, drive the 6 DC motors. Resultant force and torque generated by the DC motors are applied to the user via the probe. The generated force and the user's resistance force reach an equilibrium point somewhere.

In the haptic device, the motors, encoders, D/A board and counter boards are of very high precision. The specifications of them are listed in *Appendix B*.



Active Joints: $J_{1R}, J_{2R}, J'_{2R}, J_{3R}, J'_{3R}, J_{1L}, J_{2L}, J'_{2L}, J_{3L}, J'_{3L}$
Passive Joints: $J_{4R}, J_{5R}, J_{6R}, J_{4L}, J_{5L}, J_{6L}$

Figure 2.4. Force feedback device link structure [Adachi 2003]

The schematic structure of the current haptic device is shown in Figure 2.4. The Active Joints shown in the Figure 2.4 are driven by the motors. The motor torques are passed from Active Joints to Passive Joints and finally to the grip, which can be grasped by the human user during the working process.

2.1.2 Haptic device driver

A software driver is needed to drive the haptic device. The driver has two functions:

(I) Input function: Read counter values from counter board and decode them to solve the current tool position and orientation;

(II) Output function: Accept the force and torque calculated from a specific Virtual Environment, and translate them into corresponding digital signal. The digital signal values are converted to analog values in D/A board. The analog values are amplified by the amplifiers to drive the motors.

In order to develop the software driver, the link structure of the haptic device needs to be narrated as follows.

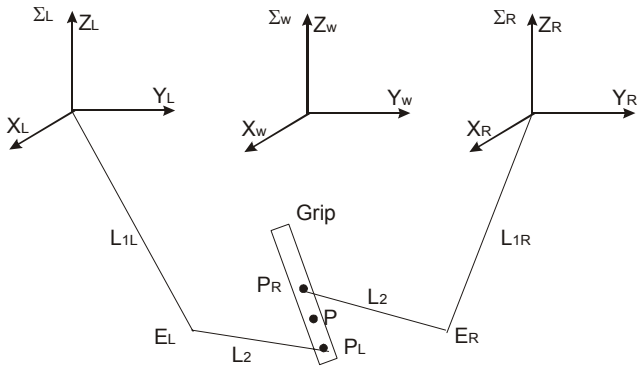


Figure 2.5. Haptic device coordinate systems definition [Adachi 2003]

Haptic device linkage

The haptic device's coordinate systems are defined as shown in Figure 2.5. The base coordinate system of the haptic device is denoted as **ΣW**. The right manipulator's local coordinate system is denoted as **ΣR**. The left manipulator's local coordinate system is denoted as **ΣL**. The joint rotation direction and angles are defined as shown in Figure 2.6.

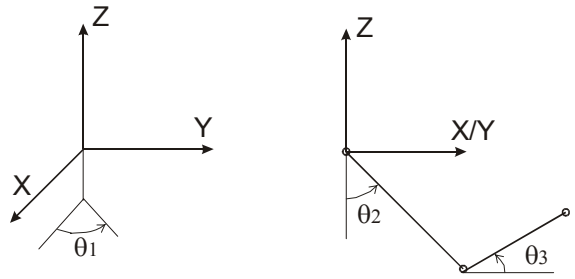


Figure 2.6. Joint rotation direction and angle definition [Adachi 2003]

Written in the base coordinate system, the end position vector of G_R is as follows,

$$r_R = \begin{bmatrix} L_{1R}S_{2R}C_{1R} + L_{2R}C_{1R}C_{3R} \\ L_{1R}S_{2R}S_{1R} + L_{2R}S_{1R}C_{3R} + L_0 \\ -L_{1R}C_{2R} + L_{2R}S_{3R} \end{bmatrix} \tag{2.1}$$

and the end position vector of G_L is as follows,

$$r_L = \begin{bmatrix} L_{1L}S_{2L}C_{1L} + L_{2L}C_{1L}C_{3L} \\ L_{1L}S_{2L}S_{1L} + L_{2L}S_{1L}C_{3L} - L_0 \\ -L_{1L}C_{2L} + L_{2L}S_{3L} \end{bmatrix} \tag{2.2}$$

Here S_{IR} is $sin\theta_{IR}$, C_{IR} is $cos\theta_{IR}$, and the rest may be deduced by analogy. L_0 is the distance between the origins of **ΣW** and **ΣL** or between the origins of **ΣW** and **ΣR**. The grip or probe's base position is the middle point of G_L and G_R. Thus the pivot position of the grip is:

$$r_G = A\frac{r_L + r_R}{2} + p \tag{2.3}$$

and the orientation of the grip is

$$s_G = A\frac{r_L - r_R}{|r_L - r_R|} = A\frac{r_{LR}}{|r_{LR}|} \tag{2.4}$$

where A and p are used for transforming the real world coordinate system to the virtual reality world coordinate system.

Assume that from the virtual reality calculation, it is required that the left manipulator needs to generate force $\vec{f}_L$ and the right manipulator needs to generate

force $\vec{f}_R$, then the necessary torques generated from the motors can be calculated by the *Jacobian Matrix* (see *Appendix A*) as they have the following relation:

$$\vec{\tau}_L = J_L^T \vec{f}_L \tag{2.5}$$

$$\vec{\tau}_R = J_R^T \vec{f}_R \tag{2.6}$$

Referring Equations (A-2) and (A-6) in *Appendix A* and using Equations (2.1) and (2.2), the Jacobian matrix of $\boldsymbol{r}$ with respect to $\boldsymbol{\theta}$ is as follows,

$$J_R = \begin{bmatrix} (-L_{1R}S_{2R} - L_{2R}C_{3R})S_{1R} & L_{1R}C_{1R}C_{2R} & -L_{2R}C_{1R}S_{3R} \\ (L_{1R}S_{2R} + L_{2R}C_{3R})C_{1R} & L_{1R}S_{1R}C_{2R} & -L_{2R}S_{2R}S_{3R} \\ 0 & L_{1R}S_{2R} & L_{2R}C_{3R} \end{bmatrix} \tag{2.7}$$

$$J_L = \begin{bmatrix} (-L_{1L}S_{2L} - L_{2L}C_{3L})S_{1L} & L_{1L}C_{1L}C_{2L} & -L_{2L}C_{1L}S_{3L} \\ (L_{1L}S_{2L} + L_{2L}C_{3L})C_{1L} & L_{1L}S_{1L}C_{2L} & -L_{2L}S_{2L}S_{3L} \\ 0 & L_{1L}LS_{2L} & L_{2L}C_{3L} \end{bmatrix} \tag{2.8}$$

where S donates *sin*, C donates *cos*, L_0 = 80mm, L_{1L} = 180mm, L_{1R} = 150mm, L_{2R} = L_{2L} = 180mm for our haptic device structure, as indicated in Figure 2.4. These are the Jacobian Matrices for the left and right manipulator, respectively. They are used for calculating the motor torques required to generate the objective torque and force on user's hand.

Principle of the driver input function

The encoder's value can be read in through the library function provided by the counter board vendor. The counter board has quadrature measurement function and the resolution of the encoder is 2000 / quadrant. Hence for a whole revolution, the resolution of the encoder is 2000 * 4 = 8000. This resolution is further improved by cog-pairs. For four of the six encoders, there are 12 / 72 cogwheel-pairs. It means in this pair of cogwheels, one cogwheel has 12 cogs and the other has 72 cogs. Hence the real resolution for detecting the grip position is 8000 * 72 / 12 = 48000. The other two encoders, which measure the Z axis rotation angles, have 12 / 105 cogwheel-pairs. It means in this pair of cogwheels, one cog has 12 cogs and the other has 105 cogs.

Hence the real resolution for detecting the grip position around Z axis is 8000 * 105 / 12 = 70000. In other words, one revolution of 360 degrees is represented with 70000 steps (Figure 2.6). Hence the relation between the counter values *counter[i]* with the joint angles θ_i of the haptic device arms are as follows (Figure 2.6):

$$\theta_i = counter[i] / steps[i] * 360 \tag{2.9}$$

where step[i] is the resolution steps for each encoder as calculated in the previous paragraph. These rotation angles are then inserted into Equations (2.1) and (2.2) to get the coordinate of the end position vector of G_R and of G_L. The consequent Equations (2.3) and (2.4) give the current position and orientation of the probe.

Before the haptic interaction process, the 'home' position of the haptic device probe needs to be registered. It is similar to NC machining, before which the spindle head needs to be homed and tool tip position has to be specified explicitly. In our haptic applications, the home position is defined as (see Figure 2.7):

$$\theta_2 = \theta_3 = 0 \tag{2.10}$$

and (see Figure 2.7)

$$\theta_1 = \arcsin(L_0 / L_2) \tag{2.11}$$

Thus, the corresponding digital value to θ_1 can be calculated as: $\theta_1 * 70000/(2\pi)$. For this haptic device, this value turns out to be 5133. When the home position is being registered, the haptic probe is lifted to the defined position and accordingly counter values are preset to be [0, 0, -5133, 0, 0, 5133], which correspond to 6 encoders, respectively.

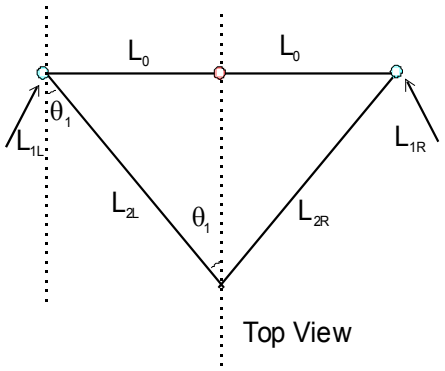


Figure 2.7. Define the home position of the haptic device

Principle of the driver output function

From the rotation angles got from function 1), the values of the current Jacobian matrix can be computed by using Equations (2.7) and (2.8). Given the required force values from the Virtual Environment, the required torques for motors can be calculated by using Equations (2.5) and (2.6).

The calculated torque values are then converted to corresponding digital values, which is used to command the D/A board. The relational coefficients between the torque values and their corresponding digital values are almost linear, according to the property experiments on the motors and amplifier. These linear coefficients are determined in the earlier experiments conducted by the Virtual Reality group at *Suzuki*, *Inc*. (Japan) [Adachi 2003].

2.1.3 General haptic rendering software

Computer Haptics is generally referred as *Haptic Rendering*. In this sub-section, three main issues are discussed: *object modeling*, *collision detection and response*, and *haptic interface stability*.

Object Modeling

The internal representation of the models in the haptic application could be polygonal surface, implicit surface, free form surface or Volume model. Polygonal surface model is the most widely used object modeling method in haptic applications. Most collision detection algorithms assume the input to be polygonal soup [Gottschalk 1996, Ho 2001, Klosowski 1998]. Implicit surface is so used because it is easier to decide whether two objects are in contact. However, its application is limited since it is not always easy to get implicit representation of surfaces. Free form surface manipulation is the desired objective for CAD/CAM since free form surface is more often used in design and manufacturing. But apparently it requires much more computation efforts. A haptic rendering method for sculptured surface models has been developed at the University of Utah [Thompson 1997]. Voxel model is the most

common Volume model [McNeely 1998]. It is used in commercial *PHANToM* system. Voxel model represents a model by 'volume pixel'. Dexel model is another kind of Volume model. It is used widely in NC simulation and verification [Huang 1994]. More details of Voxel model and Dexel model are to be presented in the next section and also in Chapter 3.

Subdivision Modeling and *Physically-based Modeling* are also involved in the object modeling in the *Computer Haptics* community. By *Subdivision Modeling*, a polyhedral model can approximate a free form surface with bounded error [Zorin 2000]. *Physically-based Modeling* introduces the physics of the reality into the Virtual Environments and thereby makes the virtual world more realistic [Baraff 1997, Qin 1996]. As a simple example, *Hooke's law* can be used for calculating the force based on the penetration depth. The research related to these two Modeling methods has many applications, like in virtual sculpting, surgical simulation, medical training and so on [Basdogan 2001, Dachille 2001, McDonnell 2000].

More discussion about object modeling is presented in Sections 2.2 and 2.3 and also throughout Chapters 3, 4 and 5.

Collision Detection and Response

Collision detection has been the focus of much research in the *Computer Graphics* community. It also plays a central role in the *Computer Haptics*. The high update rate (1 kHz or higher) largely depends on how fast the collisions among the objects are detected.

Several fast collision detection methods have been presented in the last ten years. The idea behind all these methods is *Spatial Decomposition* or *Bounding Volume*. *Spatial Decomposition* is to subdivide the space into sub-spaces. A *Bounding Volume* is to approximate the shape for each object. The object is enclosed in its bounding volume. The collision is first queried between the bounding volumes and if they overlap, further query is pursued between the objects *per se*. Therefore the number of pairs of objects to be checked for contact is greatly reduced.

Octrees, *k*-d trees, BSP-trees, BRep-indices, tetrahedral meshes, and (regular) grids are all examples of *Spatial Decomposition* techniques. By dividing the space occupied by the objects, one needs to check for contact between only those pairs of objects (or parts of objects) that are in the same or nearby cells of the decomposition. Using such decompositions in a hierarchical manner (as in Octree, BSP-trees, etc.) can further speed up the collision detection process. [Klosowski 1998a, Klosowski 1998b]

By building hierarchies of bounding volumes, one can obtain increasingly more accurate approximations of the objects, until the exact geometry of the object is reached. The choice of bounding volume could be Spheres [Hubbard 1996], Axis-Aligned Bounding Boxes (AABBs), Oriented Bounding Box (OBB) [Gottschalk 1996, Lin 1993], *k*-Discrete Orientation Polytopes (*k*-DOP) [Klosowski 1998a, Klosowski 1998b]. OBBtree, which is based on OBB, was implemented by Gottschalk et al at University of North Carolina at Chapel Hill. BVtree, which is based on *k*-DOP, was implemented by Klosowski et al at State University of New York at Stony Brook. The algorithms assume the input to be two groups of polygon (triangle) soups. The output is pairs of triangles that are involved in collision. A brief comparison of Collision Detection software is available at [Seattle Web].

In this research, new Collision Detection and Response methods are developed for haptic sculpting system and machining planning system respectively. More details about the methods are presented in Chapters 3 and 4.

Haptic Interface Stability

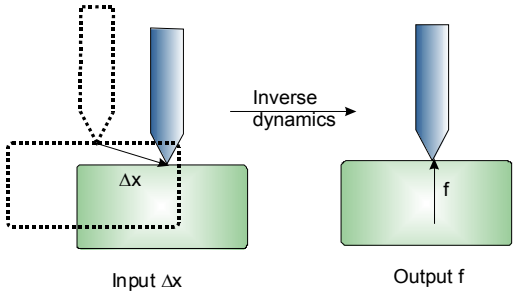
When a human user is using a haptic device in a Virtual Environment, the force feedback may be unstable. There could be chatter / vibration. Historically the stability of haptic interface has attracted a lot of attention since its initial development. The haptic device mechanical structure, haptic rendering software and human response constitute a human-computer-haptic system, in which different factors affect the stability of the force feedback. Since haptic device is a kind of robotic arm, earlier study on the stability problem can be found in literature of robotics about force

control [Craig 1989]. The stability discussion in haptic application is mostly in 1-DOF [Colgate 1994, Gillespie 1996, Hollerbach 2000].

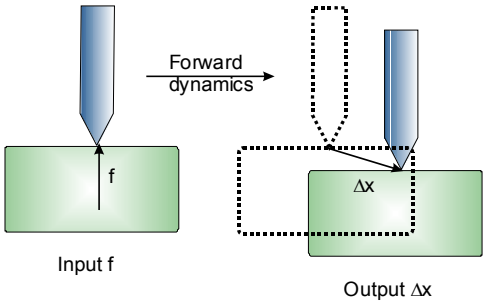
There are two general models for haptic interactions with surfaces [Hollerbach 2000]:

1) Stiffness model [Figure 2.8(a)]: the haptic interface measures displacements Δx and the simulation returns forces $f = K\Delta x$. Haptic interfaces that are good force sources are suited for implementing this mode. Our haptic device can use this model.

2) Compliance model [Figure 2.8(b)]: the haptic interface measures forces f, and simulation return displacements $\Delta x = (1/K)f = Cf$, where the inverse of the stiffness $C = 1/K$ is the compliance. Stiff haptic devices, such as industrial robots with a wrist force / torque sensor, operate in this mode.

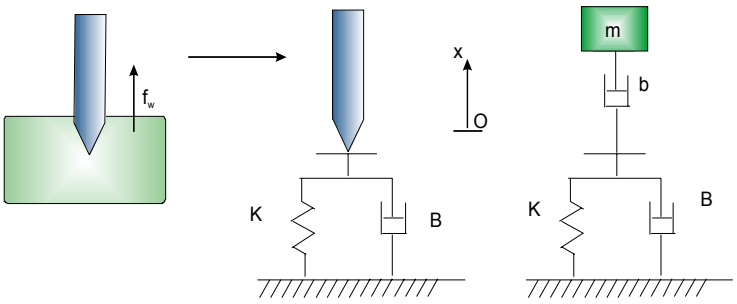


(a) Stiffness model



(b) Compliance model

Figure 2.8. Two models for haptic interactions with surfaces [Hollerbach 2000]



(a) Rigid surface model with stiffness K (b) Haptic interface model and viscosity B

Figure 2.9. Haptic interface model with rigid wall surface model [Colgate 1994]

The fundamental problem in stability is to model a rigid wall. Usually a rigid wall is modeled as a visco-elastic element, with stiffness K and viscosity B. Assuming that $x < 0$ is outside the wall and otherwise is in contact with the wall, the wall force on the haptic interface is [Figure 2.9(a)]:

$$f_w = \begin{cases} -Kx - B\dot{x} & for \quad x \le 0 \\ 0 & for \quad x > 0 \end{cases} \tag{2.12}$$

It is desirable to raise the stiffness K as high as possible for the wall to be stiff. However, if K is too high, the system may oscillate uncontrollably and seem to gain energy. This problem is analyzed and experimented by [Colgate 1994]. In [Colgate 1994], in addition to a wall model, the 1-DOF haptic interface is modeled as a mass m and damper b, pushed on by a human user [Figure 2.9(b)]. Assuming T is the sampling time of the haptic controller, the conservative criterion for the stable interfacing is:

$$b > \frac{KT}{2} + |B| \tag{2.13}$$

If the above criterion is met, the haptic interface does not gain energy from the wall interaction, and the system is stable. Two conclusions can be made from it [Hollerbach 2000]:

- Some physical damping b in the haptic interface is needed to render stiff surface. The bigger is b, the larger the surface stiffness K can be.
- The smaller the sampling period T, the larger K can be.

In [Lawrence 1996], based on the human user's psychological factors for crispness, a concept called *perceptual hardness H* is defined as the ratio of the initial rate of force change to velocity:

$$H = \frac{\dot{f}_w}{v} \tag{2.14}$$

Although H has the same units as stiffness K, perceptual hardness emphasizes how fast force changes with velocity during the initial impact transient.

Human user also plays an important role in the stability of haptic interface, since human is also a dynamic system. A haptic system designed to be stable, for example, a 1-DOF haptic interface which met the criterion (2.13), may become unstable when a human user is operating it. In [Gellespie 1996], one approach is proposed to include human user in the haptic system design. By doing so, they managed to minimize the energy leak due to the discretized sampling of the haptic interface motion.

The above discussion is limited to immobile surfaces so far. There are many cases when dynamic Virtual Environment is more attractive, such as rigid-body dynamics, deformable materials, medical simulations, or computational fluid dynamics, *etc*.

Stability problem of haptic interface has been researched for years and it turns out to be a difficult problem as involves many factors including haptic software controller, haptic device linkage and also the human user. Also the research is mainly on 1-DOF haptic interface. More research on more DOF is desirable.

2.2 Virtual Sculpting

In virtual sculpting, it usually at least has two components: a stock object to be sculpted and a sculpting tool. Theoretically there are two kinds of virtual sculpting systems based on the stock object modeling. One is based on deformable object modeling, which may be vertex-based, spline-based, particle-based, or FEM (finite element modeling)-based, *etc* [Basdogan 2002, Dachille 2001]. The other one is

similar to NC simulation process, where stock material is removed whenever a virtual tool sweeps a stock [Gaylean 1991, Leu 2002, Wang 1995]. This approach is actually based on Volume Graphics, in which an object is represented with many building blocks.

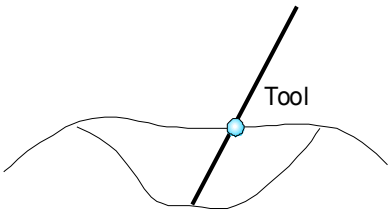
For the sculpting tools, they are usually defined with implicit surfaces with well-defined surface geometry. During the virtual sculpting process, a tool moves across the space and update the stock object. In Volume Graphics, the building blocks are either removed or altered in length. In deformable object modeling method, the surface parameters are changed and the surface shape is changed accordingly. In this research, we adopted the Volume Graphics approach, since it is natural for us to adopt the NC simulation technique. Next both approaches are reviewed.

2.2.1 Deformable object modeling

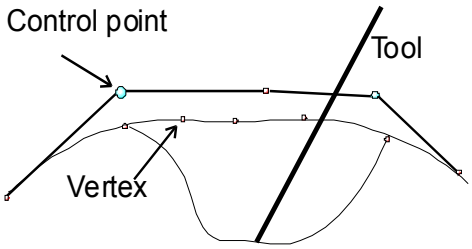
Deformation techniques can be categorized according to the approach followed to deform the surfaces: geometric or physically-based deformations [Basdogan 2002]. In geometric deformations, the object or the surrounding space is deformed based purely on geometric manipulations. Generally, a human user can manipulates vertices or control points of the 3D object surface to modify the shape of the object. Physically-based deformation techniques model the physics involved in the motion and dynamics of interactions. Models simulate physical behavior of objects under the external and internal forces. Geometric-based deformation techniques are faster but they do not simulate the underlying mechanics of deformation. On the other hand, physically-based deformation is too sophisticated for interactive, real-time simulation of multiple objects in virtual environments. The following table shows the category of deformable object models.

Table 2.1. Deformable object modeling methods [Basdogan Web]

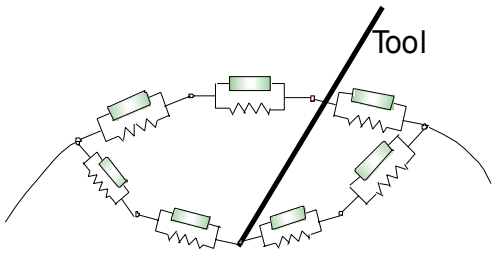
Category	**Method**	**Characteristics**
Geomtery-based	Vertex-based	Fast
	Spline-based	Smooth
Physically-based	Particle-based	Easy to implement
	Finite element based	Comprehensive, slow
	Meshless method	Finite spheres method



(a) Vertex-based method



(b) Spline-based method



(c) Particle-based method

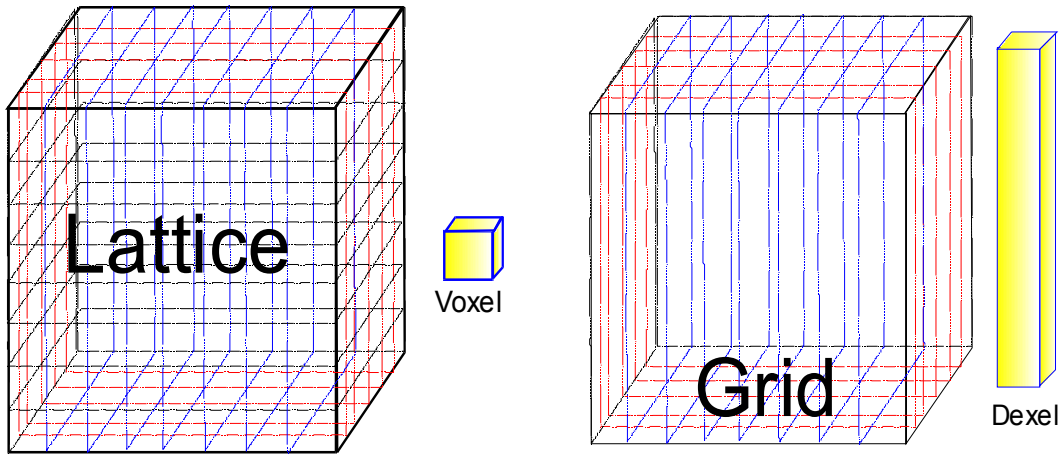
Figure 2.10. Some deformable object modeling methods [Basdogan Web]

In *Vertex-based* method [Figure 2.10(a)], the vertices of the object are manipulated to display the visual deformations. In *Spline-based* method [Figure 2.10(b)], control points of the spline surface are manipulated to achieve smooth deformations. *Particle-based* system [Figure 2.10(c)] consists of a set of point masses,

connected to each other through a network of springs and dampers, moving under the influence of internal and external forces. In this model, each particle is represented by its own mass, position, velocity and acceleration. *Particle-based* modeling is used in simulation of soft tissue and cloth behavior. *Finite element based* model divides a 3D objects into finite elements, which are assembled together to study the deformation states for the given loads. *Meshless* method is a relatively new method in physics-based soft tissue simulation [ACM SM 2001]. Recent research shows the combination of the above methods in object modeling. For example, SUNY-Stony Brook [Dachille 2001, McDonnell 2000] presented their work on spline-based physics-based geometric design, using a 3-DOF *PHANToM* interface.

2.2.2 Volume Graphics

The other kind of virtual sculpting is based on *Volume Graphics* [Kaufman 1993]. In Volume Graphics, objects are represented with small building blocks. The building blocks could be either *Voxel* or *Dexel*. Voxel model represents an object with many small cubes in a regular lattice (see Figure 2.11(a)). Dexel model represents an object with a grid of long columns compacted together (see Figure 2.11(b)). The intractable memory requirement of Voxel model makes way for the more memory-economic Dexel model in our implemented Dexel-based haptic sculpting system.



(a) Voxel model (b) Dexel model

Figure 2.11. The comparison between Voxel model and Dexel mode

Research on *Volume Graphics* has been there for decades [Kaufman 1993]. In [Gaylean 1991], Voxel-based virtual sculpting was reported with simple tools, small voxel models and also a primitive force feedback device. [Wang 1995] reported their work in Voxel-based volume sculpting. In their sculpting process, the complex operations between 3D tools and 3D stock object models are reduced down to primitive voxel-by-voxel operations. It is a difficult problem to realize interactive display for big voxel models since a voxel model may easily contain millions of voxels. They used localized ray casting image updating for interactive display during the sculpting process.

Voxel model requires a large memory to meet the space requirement of a medium size model as it is basically a 3D lattice. On the other hand, Dexel model records lists of dexels on a regular 2D grid. Hence Dexel model is more economic in memory and its rendering is also faster since the number of Dexels is much smaller compared to Voxel models of similar density. Dexel model was originally used in NC simulation and verification processes, when Van Hook used it for 3-axis NC simulation [Van Hook 1986]. More detailed work on Dexel-based NC simulation was reported in [Huang 1994]. Recently [Leu 2002] reported a Dexel-based virtual sculpting system with a 3-DOF haptic interface. Their system models the tool as a cylinder. As the cylinder is sweeping across the space and interferes with the stock object, the Dexel stock object's shape is altered with the graphics display showing the material is removed. The tool's motion forms swept volume, which is to be detailed in the next sub-section.

2.2.3 Swept volume

As a virtual tool moves across the space, it forms a swept volume [see Figure 2.12] [Chiou 1999, Blackmore 1997]. [Abdel-Malek 2003] is a good summary on their years of effort in research of swept volumes. A swept volume is defined as the volume generated by the motion of an arbitrary object along an arbitrary path (or even a surface) possibly with rotations.

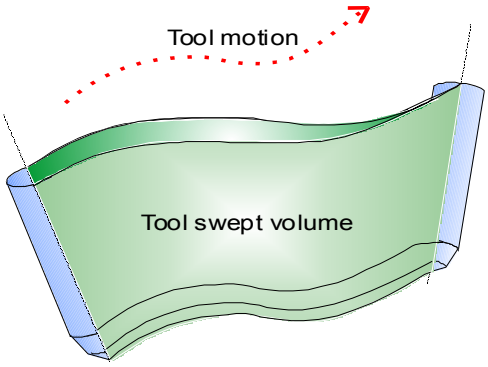


Figure 2.12. Tool swept volume

The problem of swept volume determination is concerned with the calculation of properties associated with the resulting sweep, including the delineation of the boundary for visualization and computer graphics, formulations for modeling the sweep procedure, and the calculation of the volume's mass properties for solid modeling [Abdel-Malek 2003]. In this research, an analytical methodology is used to approximate the swept volume of a tool. More details about it are presented in Chapter 3 as the underlying methodology for haptic sculpting system.

2.3 Polyhedral model machining planning

2.3.1 Polyhedral surface model

A polyhedral model usually consists of a collection of polygons. A polygon is a closed plane bounded with *n*-sides (most likely line segments). One of the most popular representation formats is STL (Standard Transformation Language) model, whose basic elements are triangular facets. STL model can also be used for Reverse Engineering (RE), Rapid Prototyping (RP), Finite Element Analysis (FEA), *etc*. In this research, STL model is also used in the haptic-aided machining planning system described in Chapter 4. The output of the haptic sculpting system is finally output as STL model, as described in Chapter 5.

STL model files can be either in text (ASCII) or binary format. Besides some header information, a STL model file is composed of a list of all the triangles. Every

triangle facet has the coordinates of its three vertices and also the normal vector. Since a triangle is usually adjacent to several other triangles, many vertices are duplicated. Figure 2.13 shows a rendered view of shoe sole STL model with its triangles displayed.

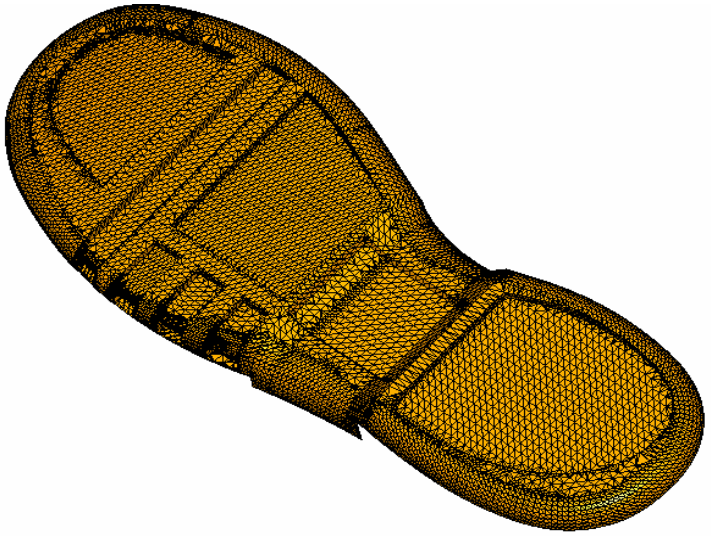


Figure 2.13. A rendered shoe sole STL model with triangles displayed

2.3.2 Machining planning

In order for a designed model to be automatically manufactured in a NC machine, machining planning has to be made to generate the NC code. Many NC *tool path generation* (TPG) methods have been proposed in the past years since the first emergence of NC machine. TPG methods are classified as either CC-based (Cutter Contact –based) methods or CL-based (Cutter Location –based) methods [Choi 1998]. In CC-based methods, tool-paths are generated by sampling a sequence of cutter contact (CC) points from the part surface and then each CC-point is converted to a CL-point. In a CL-based TPG-method, the CL-surface is used as a path-generation surface on which tool-paths are generated. The Cartesian TPG method introduced in the previous sub section can also be applied to CL-surface. *C-space* (Configuration-space) method is a generalization of this method [Choi 1998]. Compared with CC-based TPG method, the C-space approach has a number of distinctive features including: gouge- and collision-free, balanced cutting-loads, smooth cutter movements etc. It can also meet the demand of high speed machining

such as chip-load leveling, cutting-load smoothing and generation of smooth tool-paths. However, the usefulness of C-space method is rather limited in 5-axis machining. The C-space method is not easily applied to 5-axis machining with a flat or filleted endmill cutter mainly because the concept of CL-surface is not defined in this case. Hence two concepts: 'position' C-space and 'orientation' C-space have been introduced to describe the tool position and tool orientation respectively.

The most common 5-Axis *tool path generation* (TPG) method is Iso-planar method. In this method, a series of cutting planes are used to cut the part surface and thus a series of cutter contact points can be obtained. The part surface is usually triangulated into triangular facets. Therefore the CC points are just the intersection of the cutting planes and the triangular facets. The cutter orientation is then further determined. Usually the curvature matching principal is applied to match the local tool surface and the local part surface. Gouging detection and correction are further applied to find a feasible tool orientation. So far this problem is not completely solved due to its expensive computation.

Common complex surface machining errors and inefficiencies that result from NC programming methods are gouging and undercuts. When a portion of the cutting edge of the cutter extends below the surface by more than the allowable surface tolerance, gouging occurs at this position. Undercut error occurs when an excess of material is left by the finishing tool paths and grinding is required to bring the surface finish within the specified tolerance.

2.3.3 Pencil-cut

In NC machining, usually it is not expected that the part surface can be machined once and for all. Some clean-up machining needs to be run in order to remove the left material. Pencil-cut is the operation by which some materials near the shared edges are removed [see Figure 2.14]. In [Choi 1998], it was listed that pencil-cut can be used in Roughing, Semi-Finishing, Finishing and Clean-up machining stage. While in [Choi 1998], the pencil-cut is limited to 3-axis machining,

the concept can be extended to 5-axis machining. It was proposed in Chapter 4 that considering the machine dynamics, one might want to find some areas (most likely near the shared edges of surface patches) where the tool orientation angle is interpolated. Usually in these areas the tool is lifted up in order to keep the interpolated orientation angle and thus some materials are left.

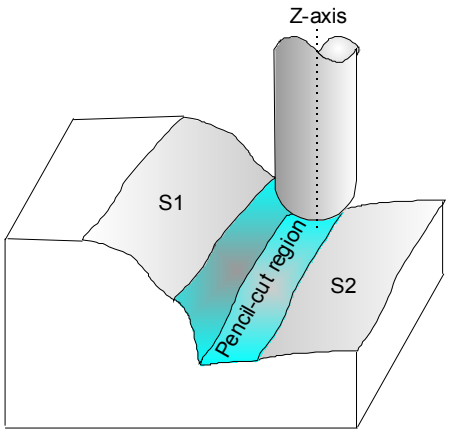


Figure 2.14. Pencil-cut in NC machining

In 5-axis machining, this kind of clean-up region can be identified by some methods, for example, Rolling Ball method [Ren 2002b] (also see Chapter 4). But again, to find a good machining strategy for Pencil-cut is not a trivial task. This machining strategy includes the tool move direction, tool orientation, step length and step-over length. Haptic interface may help. By using haptic interface and pre-identified pencil-cut region, the user can suggest some feasible cutting direction and orientation for the pencil-cut. In Chapter 4, 5-axis pencil-cut is proposed with haptic interface.

2.4 Summary

In this chapter, we reviewed the related techniques which are going to be used later in this paper. The research background of this research is also presented. As the pre-cursor of this research, the haptic device and its controller were constructed. Its working principles are described to facilitate the understanding its applications to be presented in the next chapters.

Chapter 3

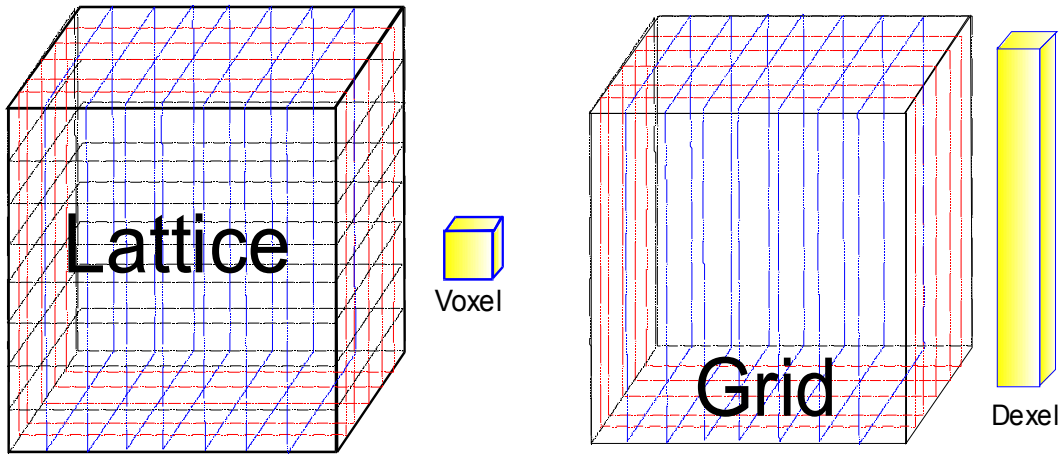
Analytical Methodology of Dexel-based Virtual Sculpting

This chapter presents an analytical methodology for virtual sculpting of complex surfaces with a developed 5-DOF (Degree of Freedom) force-torque haptic interface. In the proposed methodology, 5-axis tool motion and analytical tool swept volume are formulated for updating the virtual stock material, which is represented with Dexel volume model. Based on the tool motion analysis, a *dexel-based collision detection* method and a *force-torque feedback* algorithm are proposed for virtual sculpting. A lab-built 5-DOF haptic interface system (see Chapter 2) is developed for the proposed haptic sculpting system. With the proposed methodology, a user can virtually sculpt a volume stock to get an intuitive design by using the haptic interface. From the haptic sculpting system, both corresponding tool motion of the creative process and the sculpted model can be output. The tool motion can be recorded and output as NC (numerically-controlled) commands. The output of the haptic sculpting system can be processed for machining planning, which is presented in Chapter 4 [Zhu 2000b and Zhu 2003d]. Computer implementation of the haptic sculpting system based on the analytical methodology is also presented in this chapter.

3.1 Introduction

Virtual sculpting rose up more than a decade ago when it was presented as a novel free-form interactive modeling technique [Coquillart 1990, Galyean 1991]. There are two kinds of virtual sculpting systems theoretically. One is derived from surface-based deformable geometric object modeling, which may be vertex-based, spline-based, particle-based, FEM (finite element modeling)-based, *etc*. [Basdogan Web, Dachille 2001]. The other one is analogous to NC (Numerically-Controlled) simulation process, where stock material is removed whenever a virtual tool sweeps a

stock [Leu 2002, Wang 1995]. This type of virtual sculpting is usually based on volume models, e.g., *Voxel* or *Dexel* models. *Voxel* model represents an object with many small cubes in a regular lattice (see Figure 3.1(a)). *Dexel* model represents an object with a grid of long columns compacted together (see Figure 3.1(b)).



(a) Voxel model (b) Dexel model

Figure 3.1. The comparison between Voxel model and Dexel model (same as **Figure 2.11**)

It was recognized that the feeling of actuality may be greatly enhanced with a force feedback interface. Simple earlier integration of force feedback interface with a virtual sculpting system was attempted as in [Galyean 1991]. Haptics is a novel discipline concerned with information and objects manipulation through touch and force feedback. Haptic interface provides force and/or torque feedback to the user in a virtual environment. In the recent years, commercially available haptic interface has been integrated with virtual sculpting systems. A dynamic sculpting system for free-form subdivision solids was developed in SUNY Stony Brook [Dachille 2001]. A Dexel-based virtual carving system was developed at University of Missouri-Rolla [Leu 2002]. Both systems are integrated with a 6-DOF input and 3-DOF output commercially available haptic device.

In this chapter, an analytical methodology for the newly-developed *dexel-based haptic sculpting system* is presented. A haptic sculpting system based on the analytical methodology of virtual prototyping has been developed in our lab, as

described in Chapter 2. A 6-DOF input and 5-DOF output haptic device is integrated with the developed haptic sculpting system. In the system, a user can virtually sculpt a raw stock to get an intuitive design by using a haptic interface. The designed model can be further processed for machining planning [Zhu 2000b and Zhu 2000d].

The remaining of this chapter is organized as follows. Section 3.2 narrates the methodology of *Dexel-updating process* with tool motion. Section 3.3 describes the 5-DOF Dexel-based haptic rendering (force and torque feedback) programming in details. Section 3.4 illustrates the integrated software and hardware developed in the haptic interface system. Section 3.5 is some discussion about the current methodology. Implementation and experiment results are presented in Section 3.6, followed by the concluding remarks.

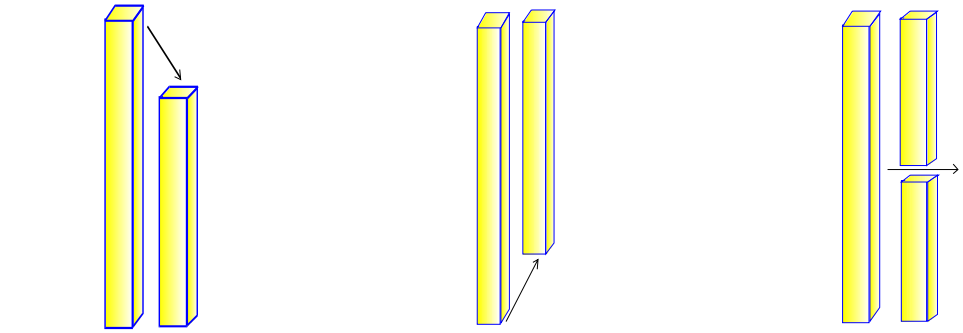
3.2 Methodology of Dexel-based Virtual Sculpting

3.2.1 Object modeling in virtual sculpting

A virtual sculpting system usually contains two basic components: a stock material to be sculpted and a virtual tool. The stock material is represented with Dexel volume model in this chapter. Dexel volume model, as shown in Figure 3.1(b) has been used widely in NC simulation process [Van Hook 1986, Huang 1994]. Compared to Voxel model (Figure 3.1(a)), which is used extensively in Volume Graphics, the advantage of using Dexel volume model is its smaller memory requirement and fewer processing volume elements. Dexel volume model represents an object with a grid of long columns compacted together (see Figure 3.1(b)). Three possible situations may happen to a Dexel element during a Dexel-updating process:

1) The top part of a Dexel element is shortened by a cutter (Figure 3.2(a));
2) The bottom part of a Dexel element is shortened by a cutter (Figure 3.2(b));
3) A Dexel element is broken into two parts if a cutter cuts the Dexel element in the middle (Figure 3.2(c)).

Therefore in the data structure, a Dexel volume model is recorded as a regular 2D grid, with each grid point associated with a linked list of Dexel elements (Figure 3.3). For simplicity, only a few sample Dexel linked lists are shown in Figure 3.3. In the Dexel-updating process, each Dexel is treated as a vector (see Figure 3.4).



(a) Shorten the top part (b) Shorten the bottom part (c) Cut a Dexel in the middle

Figure 3.2. Three Dexel-updating situations

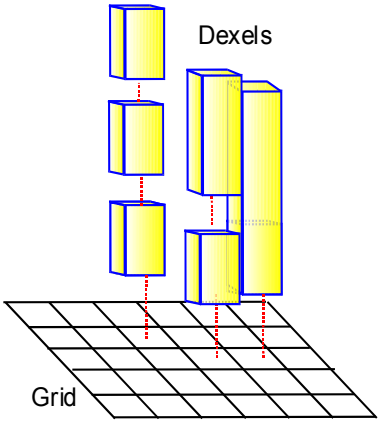


Figure 3.3. The linked list data structure of Dexel model

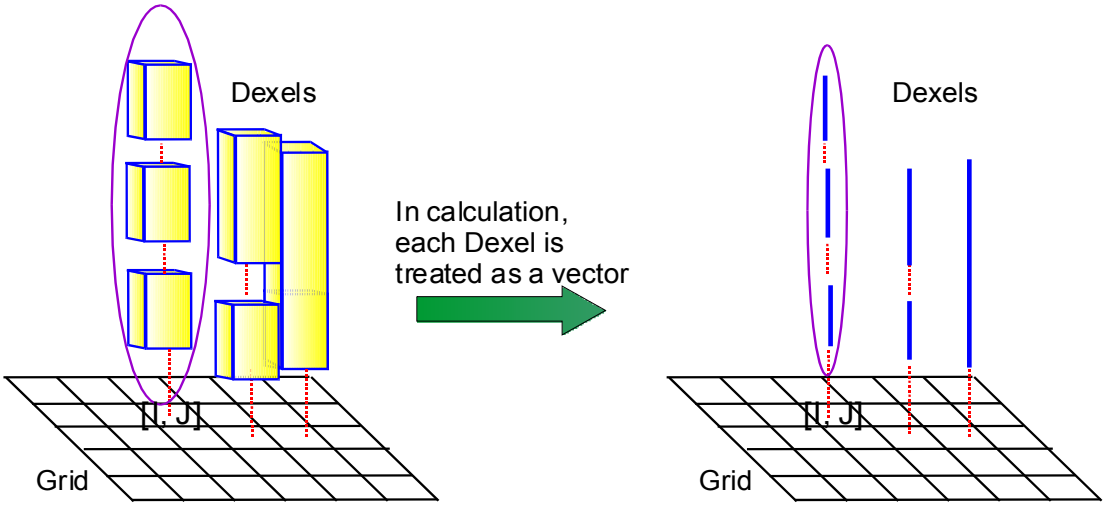
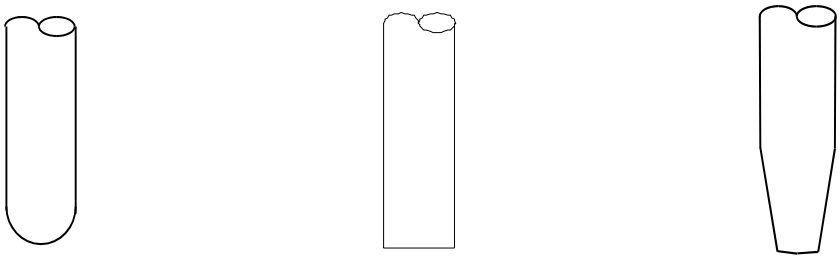


Figure 3.4. In the *Dexel-updating process*, each Dexel is treated as a vector

In this design of virtual sculpting process, the tool definition is defined as a milling cutter used in a machining workshop. The tool can be defined as combination of 2D circular arcs and straight lines (Figure 3.5). Thus it can represent ball endmill cutter, flat endmill cutter and taper endmill cutter in NC (numerically-controlled) machining (Figure 3.5). The details will be discussed in the following sections.



(a) Ball endmill (b) Flat endmill (c) Taper endmill

Figure 3.5. Common milling tools used in machining workshops

3.2.2 Fundamentals of tool swept volume and the haptic sculpting tool motion

As a tool sweeps across the space, it generates a swept volume. The requirement of *Dexel-updating process* is: given a pair of (X, Y) coordinates, calculate the Z values of tool swept volume (Figure 3.6). The general theory of swept volume is dominated by the geometric relationship defined by some differential equations [Blackmore 1997]. The computation based on the differential equations is complex and it does not apply to *Dexel-updating process* well due to the different parameterization. In the calculation of swept volume in [Blackmore 1997], the tool swept volume has to be triangulated nevertheless. In our earlier work presented in [Chiou 2002], a formulation has been proposed to find the analytical solution for the tool swept volume. This method also has similar limitation as in [Blackmore 1997]. In [Mann 2002], a generalized imprint method was used to find the general swept surface of revolution in NC machining. This imprint method connects grazing points of two adjacent tool positions with line segments and then gets approximation of the swept surface by triangulation. The method has the potential defect that two adjacent tool positions do not have the corresponding grazing points to connect. In [Huang

1994], many instances of the tool are densely arranged along the tool path to update the Dexel model.

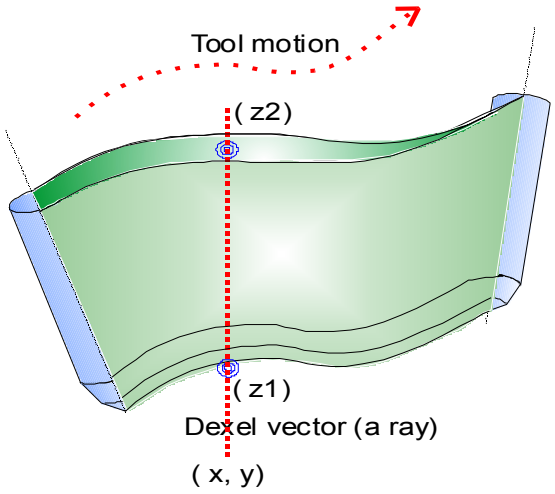


Figure 3.6. Finding intersection point of a Dexel vector (ray) with a tool swept volume

In this chapter, a method based on tool motion analysis and analytical swept volume is proposed for fast updating of Dexel volume models. The proposed *Dexel-updating* method is used for *haptic virtual sculpting* process. For virtual sculpting, some important observations are shown as follows:

- The tool motion is determined from the posture of the haptic probe. It is determined in real-time, instead of pre-determined NC code command as in NC simulation.
- The posture information of the haptic probe is discrete depending on the sample digital signal from the electronic haptic controller. Usually the haptic sampling frequency is very high (about 1000*Hz*) and thus between two sampled tool motion, the tool motion changes very little if the haptic probe does not move rapidly. Therefore the complications, such as self intersection, involved in general swept volume rarely happen between any two successive digital sampling instances.

- The task of haptic sculpting is more time-critical than NC simulation. Hence in haptic sculpting the objective is to get fast haptic rendering by finding a trade-off between updating precision and speed requirement.

Based on the above observation, the following approach is proposed. The objective of the proposed approach is to find analytical solutions for fast and stable Dexel model updating process.

3.2.3 Analytical methodology of tool motion and updating Dexel model for virtual sculpting

In this chapter, the tool motion is analyzed to find the tool swept volume. The tool swept volume is modeled and calculated to update stock material represented with Dexel volume model.

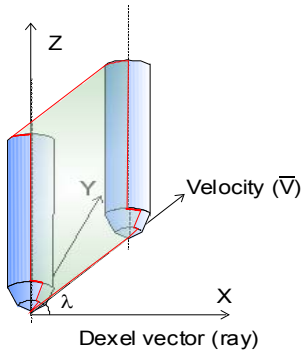


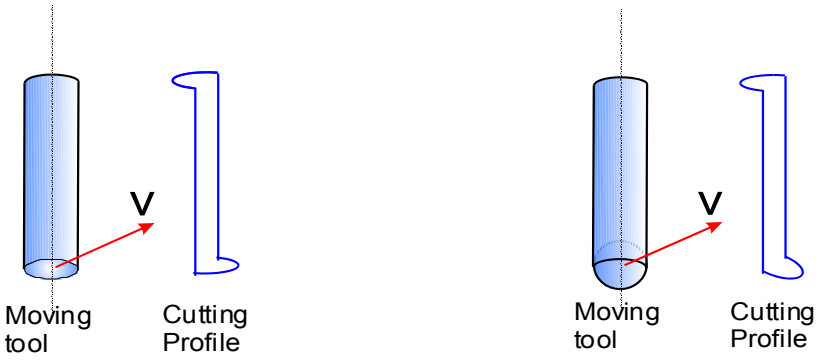
Figure 3.7. Local coordinate system setup with current tool orientation as Z axis

To get the tool swept volume, first the *tool cutting profile* (TCP) needs to be found for the tool. As presented in our earlier work in [Chiou 1999], *Tool cutting profile* is the profile curve which is the solution to the following equation:

$$\vec{n} \bullet \vec{V} = 0 \qquad (3.1)$$

where $\vec{n}$ is the tool surface normal and $\vec{V}$ is the instantaneous tool moving direction. The tool motion is discretized into linear segments. After the discretization of the tool motion, the two adjacent discretized tool orientations are linearly interpolated and a *local coordinate system* (LCS) can be setup as shown in Figure 3.7.

During the Dexel-updating process, a *local coordinate system* (LCS) needs to be setup first so that the current tool orientation is the local *Z* axis and *X* axis point from the tool tip of the first tool position to the second tool position, as shown in Figure 3.7. In Figure 3.7, $\vec{V}$ is the direction of the tool movement and is within the *X*-*Z* plane.



(a) Cutting profile for a ball endmill (b) Cutting profile for a ball endmill

Figure 3.8. Cutting profile for flat endmill and ball endmill

Using Equation (3.1), the cutting profile of a cutter moving along the vector $\vec{V}$ can be solved [Chiou 2002]. If the tool is a ball endmill (a half sphere plus a cylinder) or flat endmill cutter (a cylinder), the cutting profile can be easily found. It is composed of two half circular arcs plus two straight lines, as shown in Figure 3.8. For an endmill with (truncated) cone(s) at the bottom (e.g. taper endmill), the cutting profile of the cone portion has to be derived. In Figure 3.9, a truncated cone is defined as follows:

$$Cone = \begin{bmatrix} (r_c + l_c * \tan\beta_c) * \cos\theta \\ (r_c + l_c * \tan\beta_c) * \sin\theta \\ h_c + l_c \end{bmatrix} \tag{3.2}$$

where l_c is the height parameter of the truncated cone,

θ is the angle parameter around the Z axis,

h_c is the distance from the origin point to the cone lower end,

r_c is the radius of the lower end, β_c is the cone angle.

Its surface normal can be found as follows:

$$\vec{n}_{cone} = \frac{\partial Cone}{\partial l_c} \times \frac{\partial Cone}{\partial \theta} = \begin{bmatrix} \cos\theta \\ \sin\theta \\ -\tan\beta_c \end{bmatrix} \tag{3.3}$$

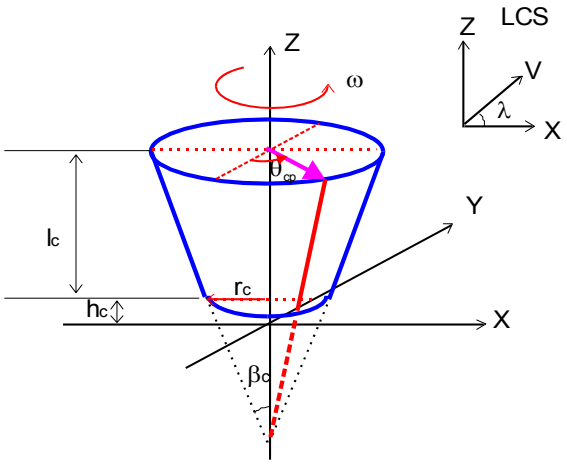


Figure 3.9. Definition of the cone of a cone endmill and the relation between θ and moving direction *V*, in order to find the tool cutting profile

If $\vec{V}$ is along the *X* axis direction, then:

$$\vec{V} = \begin{bmatrix} 1 & 0 & 0 \end{bmatrix}^T \tag{3.4}$$

Now let the cone rotate around *Y* axis for a certain angle λ, that is, the cone is moving along a direction vector which forms a λ angle with *X* axis, as shown in Figure 3.8, and then also rotate around *Z* axis for a certain angle ω, so that:

$$\vec{V} = \begin{bmatrix} \cos(\omega) & \sin(\omega) & 0 \\ -\sin(\omega) & \cos(\omega) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos(\lambda) & 0 & -\sin(\lambda) \\ 0 & 1 & 0 \\ \sin(\lambda) & 0 & \cos(\lambda) \end{bmatrix} \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} \tag{3.5}$$

By substituting Equations (3.3) and (3.5) into Equation (3.1), the following equation can be obtained:

$$\cos\omega \cdot \cos\lambda \cdot \cos\theta - \sin\omega \cdot \cos\lambda \cdot \sin\theta = \sin\lambda \cdot tg\beta_c \tag{3.6}$$

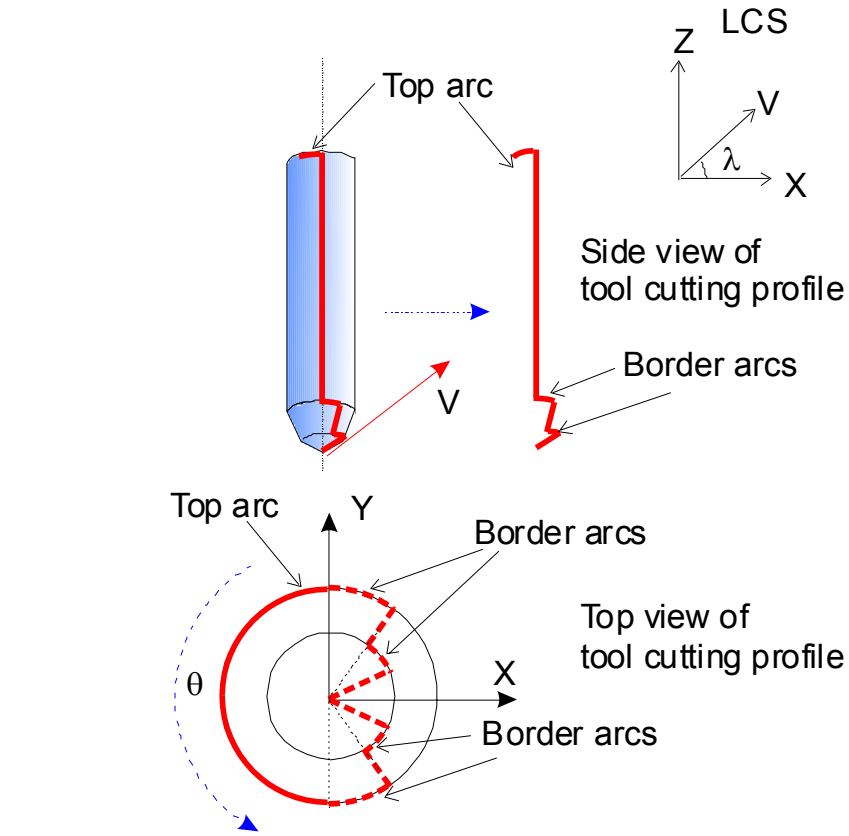
which can be reduced to the following equation:

$$\cos(\omega + \theta) = tg\lambda \cdot tg\beta_c \tag{3.7}$$

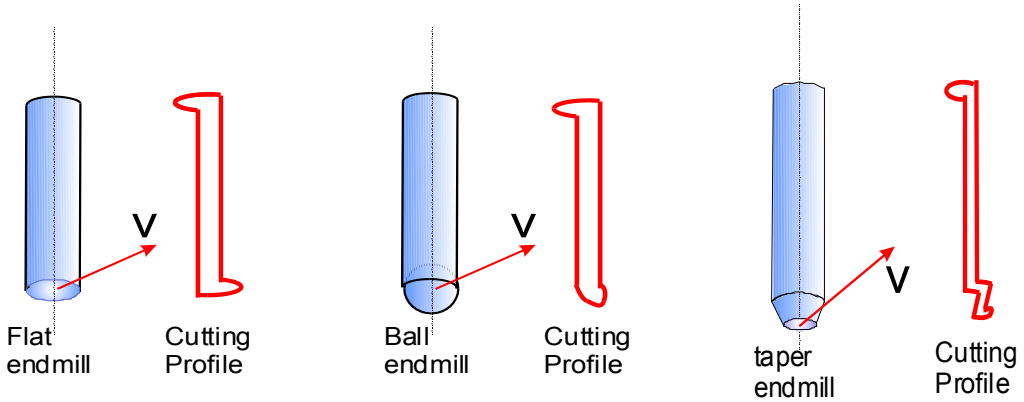
The solution of Equation (3.7) is the *cutting profile angle* θ_{cp}, which can be found as:

$$\theta_{cp} = \arccos(\tan(\lambda) * \tan(\beta_c)) - \omega \qquad (3.8)$$

This equation gives the explicit relation between tool orientation (λ, ω), cone angle β_c and cutting profile angle θ_{cp}. It means that, given a tool orientation (λ, ω), the cutting profile angle θ_{cp} can be found by Equation (3.8) (Figure 3.9). In the current *local coordinate system* as shown as in Figure 3.6, ω is always 0 as in Equation (3.8).



(a) Tool cutting profile details of a cone end tool



(b) Flat endmill (c) Ball endmill (d) taper endmill

Figure 3.10. Tool cutting profile of different kinds of tools

The above discussion narrated the calculation of *tool cutting profiles* for cylinder, sphere and truncated cones. The *tool cutting profile* of a tool composed of 2D arcs and straight lines can be derived by combining its component tool cutting profiles. For example, for a tool with two stacked cones at the bottom, the cutting profile is composed of top arc, several line segments and border arcs, as shown in Figure 3.10(a) [Chiou 2002]. Figures 3.10(b), (c) and (d) list the tool cutting profiles of different kinds of cutting tools. The movement of the cutting profiles generates a swept envelope across the space (Figures 3.6 and 3.7). The swept envelope together with two end tool shapes constitutes a swept volume as a result of tool motion (Figures 3.6 and 3.7). The *tool cutting profile* defined by Equation (3.1) is used in generating the 3D swept envelope as detailed in the following paragraphs.

In the following discussion, a generalized tool with the tool tip consisting of two cones is used as an example for illustration (Figure 3.10(a)). This tool is referred to as *cone end tool* in this chapter. In this approach, four different general tool motion situations are considered in this chapter, detailed as follows:

I.) *Tool moves up and down along the tool axis*: the intersection calculation tasks include intersection with a tool M and a cylinder, which is the swept volume $SV(M)$ for this case (see Figure 3.11):

$$SV(M) = \bigcup \{M, Cylinder\} \tag{3.9}$$

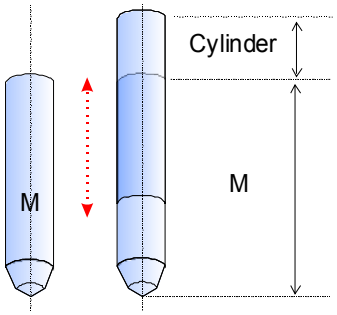


Figure 3.11. Tool move up and down along the tool axis

II.) *Tool rotates around an axis without translation* (see Figure 3.12(a)): the rotation of straight lines around an axis turns out to be cones (Figure 3.15)

and the rotation of a sphere around its center is always a ball. Take taper endmill as the example, the swept volume $SV(M)$ of tool rotation motion can be broken down to two side planes, two cones (bottom rotation) and a top cylinder (approximately) (Figure 3.12(b)):

$$\begin{aligned} SV(M) &= \bigcup \{M_i, \textit{Rotate Swept Envelope}, M_{i+1}\} \\ &= \bigcup \{M_i, \textit{Left side rotate}, \textit{Right side rotate}, \textit{Cones Rotate}, M_{i+1}\} \end{aligned} \tag{3.10}$$

The swept surface of the top circle is approximated with a partial cylindrical surface. This is an acceptable approximation since the tool tip, not the tool top, is used for sculpting. Hence it boils down to the problems of ray-cone intersection and ray-sphere intersection. Analytical solutions are not difficult to find while coordinate transformation needs to be carefully treated.

III.) *Tool translates without rotation*: the tool movement is linear and orientation is fixed, as shown in Figure 3.13. The cutting profile is utilized to calculate the swept volume. As mentioned before, for this *cone end tool*, the cutting profile is composed of top arc, several line segments and border arcs, as shown in Figure 3.10(a). Hence this tool swept volume $SV(M_i \to M_{i+1})$ of a tool M movement from M_i to M_{i+1} can be broken down into top arc swept pipe, sides, border arc swept pipes and cone swept (see Figure 3.13).

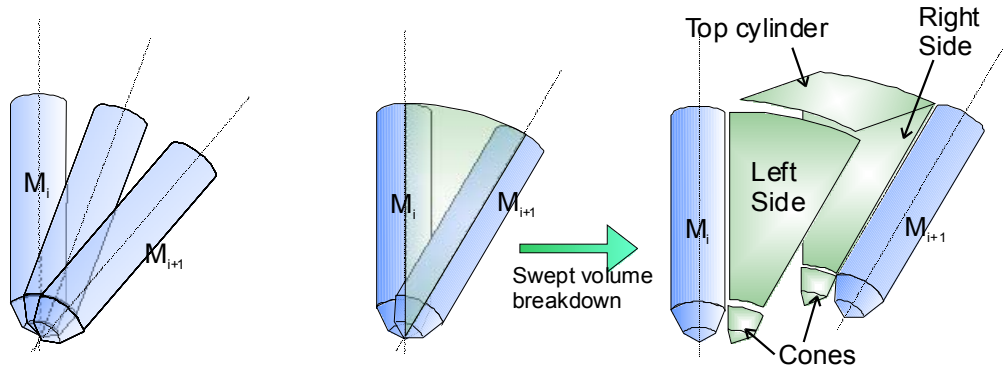
$$\begin{aligned} SV(M_i \to M_{i+1}) &= \bigcup \{M_i, \textit{Translate Swept Envelope}, M_{i+1}\} \\ &= \bigcup \begin{Bmatrix} M_i, \textit{Left side}, \textit{Right side}, \textit{Top swept pipe}, \\ \textit{Border swept pipes}, \textit{Cones swept}, M_{i+1} \end{Bmatrix} \end{aligned} \tag{3.11}$$

IV.) *Tool moves around with both rotation and translation*: this is the most common case during virtual sculpting. In this case, the tool movement is discretized into small linear segments as shown in Figure 3.14. The difference between each adjacent tool orientations is limited to a small increment angle. Inside each such discretized segment $SV(M_i \to M_{i+1})$, the tool movement is considered to be linear and orientation is linearly

interpolated. Thus the tool swept volume $SV(M)$ of a tool M movement from M_1 to M_n is the union of $(n\text{-}1)$ discretized sub swept volume $SV(M_i \rightarrow M_{i+1})$:

$$SV(M) = \bigcup_{i=1}^{n-1} SV(M_i \rightarrow M_{i+1}) \tag{3.12}$$

where $SV(M_i \rightarrow M_{i+1})$ is defined as in Equation (3.11).



(a) tool rotates only (b) tool swept volume breakdown

Figure 3.12. Tool rotates only, without translation

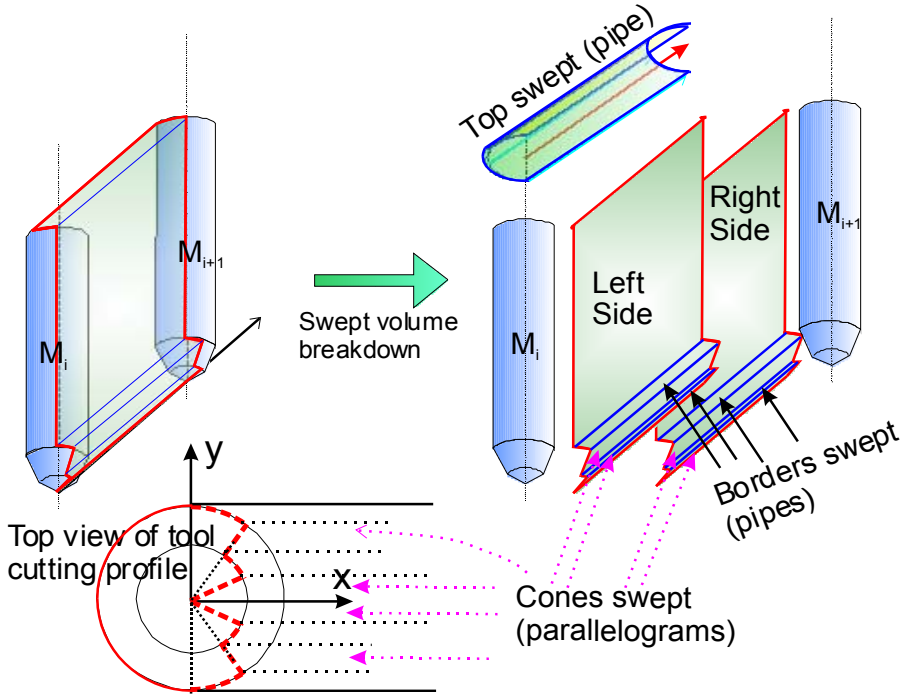
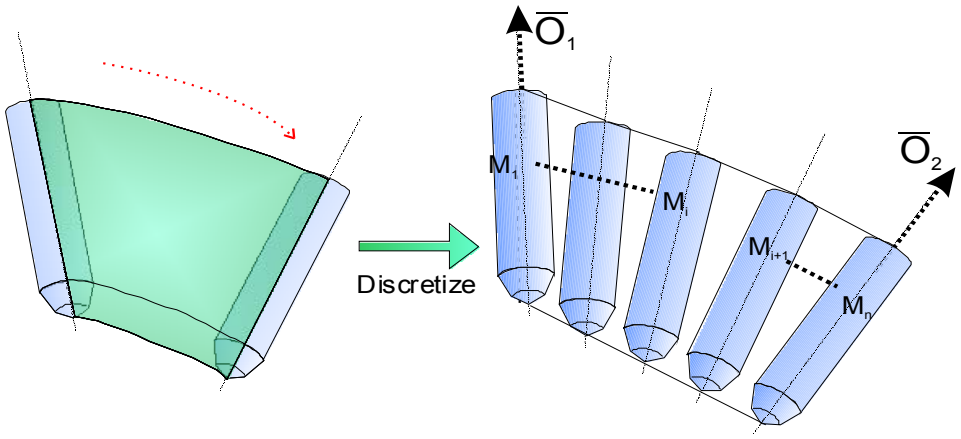


Figure 3.13. Tool moves linearly without rotation: tool swept volume breakdown



Tool rotates and translates at the same time:
discretize tool motion and process each linear segment

Figure 3.14. General tool motion discretization during the virtual sculpting process

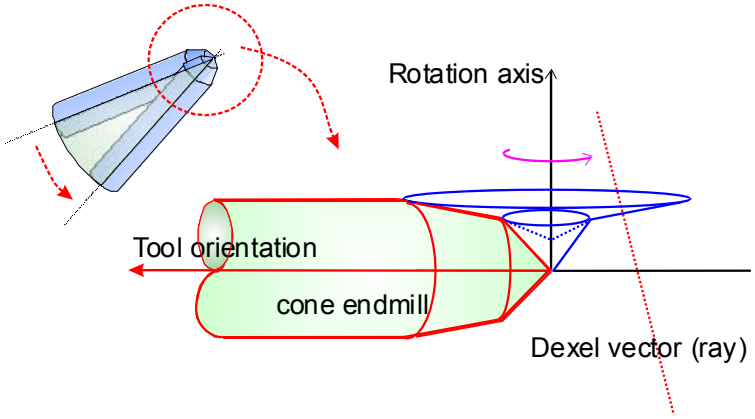


Figure 3.15. Intersection of a cone with a transformed Dexel vector during a tool's rotation

After the tool swept volume $SV(M)$ is identified by the above analysis, it is used to update the stock material represented with Dexel volume models. As mentioned before, task for *Dexel-updating process* is: given a pair of (X, Y) coordinates, calculate the Z values of tool swept envelope (Figure 3.6). The advantage of the current approach is that all the intersection calculation can be found from analytical solutions and thus any numerical iteration is avoided.

During the tool interference tests, coordinate transformations are heavily involved to facilitate the calculation of the intersection problem. According to the above *tool cutting profile* and tool motion analysis, the intersection between the tool

swept volume and the Dexel vectors breaks down to several basic intersection problems (also see Figures 3.15 to 3.17):

1) *Intersection between a tool and a Dexel vector (ray)*: this is to find the intersection of Ray-Cylinder, Ray-Cone and/or Ray-Sphere. The derivation is not difficult and there are available algorithms in [Held 1997].
2) *Intersection between a cone end tool rotational swept surface and a Dexel vector (ray)*: as mentioned above, Ray-Cone intersection can also be used during the rotation of a tool with cones as the tool tip, e.g. taper endmill or a *cone end tool*. In such situation, a straight line rotates around an axis to form a partial cone (Figure 3.15).
3) *Intersection between a parallelogram and a Dexel vector (ray)*: as shown in Figure 3.16, the parametric representation of a parallelogram can be as follows:

$$P = b(P_3 - P_1) + a(P_2 - P_1) + P_1 \tag{3.13}$$

where a and b are two parameters. Since (x, y) coordinate of point P is given, the parameter a and b can be solved (see Figure 3.16) and consequently the Z coordinate of intersection point P can be found.

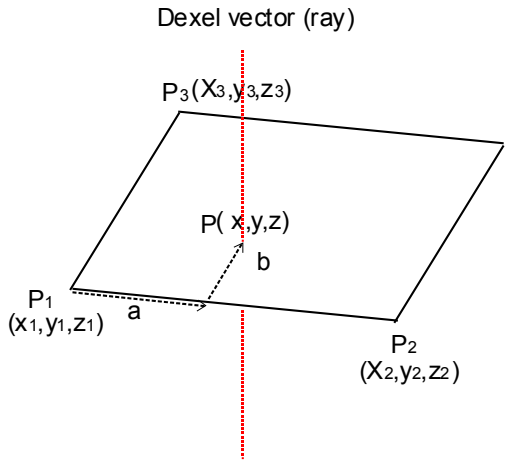


Figure 3.16. Parameterization of a parallelogram and intersection with a vertical Dexel vector

4) *Intersection between a pipe and a Dexel vector (ray)*: as shown in Figure 3.17, a 2D planar circular arc can sweep along a straight line to form a ruled surface with the shape of a partial pipe. Assume that the circular arc with radius r lies in the X-Y plane with the center at ($0,\ 0,\ z_h$) and the arc is moving along the direction $\vec{V}$ ($V_x,\ 0,\ V_z$), as shown in Figure 3.17. The 3D ruled surface generated by a planar circular arc's linear movement may be represented in the local coordinate system as follows (Figure 3.17):

$$Pipe = \begin{bmatrix} x \\ y \\ z \end{bmatrix} = arc + s * \vec{V} = \begin{bmatrix} r * \cos\alpha \\ r * \sin\alpha \\ z_h \end{bmatrix} + s * \begin{bmatrix} V_x \\ 0 \\ V_z \end{bmatrix} = \begin{bmatrix} r * \cos\alpha + s * V_x \\ r * \sin\alpha \\ z_h + s * V_z \end{bmatrix} \tag{3.14}$$

where s is the parameter of tool moving direction vector,

α is the angle parameter of 2D planar circular arc.

A Dexel vector (ray) may be represented in the local coordinate system (Figure 3.17) as:

$$Q = \begin{bmatrix} x \\ y \\ z \end{bmatrix} = P_0 + t * \vec{W} = \begin{bmatrix} P_{0x} \\ P_{0y} \\ P_{0z} \end{bmatrix} + t * \begin{bmatrix} W_x \\ W_y \\ W_z \end{bmatrix} = \begin{bmatrix} P_{0x} + t * W_x \\ P_{0y} + t * W_y \\ P_{0z} + t * W_z \end{bmatrix} \tag{3.15}$$

where $P_0(P_{0x},\ P_{0y},\ P_{0z})$ is the start point of a Dexel vector,

t is the parameter of the Dexel vector,

$\vec{W}(W_x, W_y, W_z)$ is the direction cosine of the Dexel vector ray.

To find the intersection point between the pipe and the Dexel vector ray, the following equation is needs to be solved:

$$Pipe = Q \tag{3.16}$$

By solving Equation (3.16), parameter t can be obtained from the following quadratic equation:

$$\left(b_1^2 + W_y^2\right)t^2 + 2\left(a_1 b_1 + P_{0y} W_y\right)t + \left(a_1^2 + P_{0y}^2 - r^2\right) = 0 \tag{3.17}$$

where $a_1 = P_{0x} - \left(P_{0z} - z_h\right) \cdot \left(\frac{V_x}{V_z}\right)$, $b_1 = W_x - W_z \cdot \left(\frac{V_x}{V_z}\right)$.

The other two parameters *s*, α can calculated accordingly and finally the intersection point coordinates can be obtained.

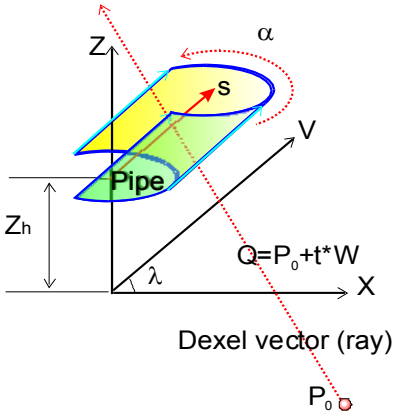


Figure 3.17. Finding the intersection between a pipe and a Dexel vector (ray)

The above analysis is used for updating Dexel volume model. According to the analysis, no triangulation of the swept volume is needed and analytical solution can be found for all related component problems, which include ray-cylinder, ray-cone, ray-sphere, ray-parallelogram and ray-pipe intersections. In the next section, a Dexel-based 5-DOF force-torque feedback algorithm is proposed based on the previous analysis for the virtual sculpting.

3.3 Five-DOF Haptic rendering programming for virtual sculpting

Haptic rendering refers to the task of detecting interference between objects in a virtual environment and accordingly calculating the force and/or torque feedback. The common requirement for haptic feedback rate is 1000*Hz* in order to sustain a continuous force feedback. When this rate cannot be obtained due to the limitation of computation power, interpolation of force data may be pursued to meet the 1000*Hz* rate requirement. Unlike a common 3-DOF force feedback, where intermediate

representation [Adachi 1995] and 'god object' [Zilles 1995] are usually used, 5-DOF dexel-based collision detection and force-torque feedback is completely different. In this chapter, the force feedback is formulated based on the material removal at a given instance. The force can be considered to be proportional to the *Material Removal Rate* (MRR).

In this chapter, a *Dexel Removal Rate* (DRR) approach is proposed to analyze the haptic sculpting force-torque feedbacks. MRR can be considered to be proportional to the dexel volume removed accumulatively per unit time, which is defined by us as *Dexel Removal Rate* (DRR). The haptic force response is calculated based on the DRR that is proportional to the dexel volume removal. Based on the analysis from the previous section, the DRR can be calculated during the *Dexel-updating process*.

As shown in Figure 3.18(a), a component force magnitude $f_{i,mag}$ can be defined as proportional to the removed dexel volume and is given by the following formula:

$$f_{i,mag} = k \cdot (\Delta x_i \cdot Dx_{area}) = (k \cdot Dx_{area}) \cdot \Delta x_i = k' \cdot \Delta x_i \tag{3.18}$$

where Δx_i is the height of the dexel segment,

Dx_{area} is the cross section of the Dexel elements,

k is the pre-defined cross-section dependent force coefficient,

k' is the cross-section independent force coefficient.

Although similar in form, this is NOT *Hooke*'s law. It is based on the assumption that the force is proportional to the removed dexel volume. The force feedback direction $\vec{f}_{i,dir}$ is opposite to the 'tool-approaching direction' to each removed dexel volume.

The torque calculation method is shown as in Figure 3.18(b) and $\vec{\tau}_i$ is calculated by the following formula:

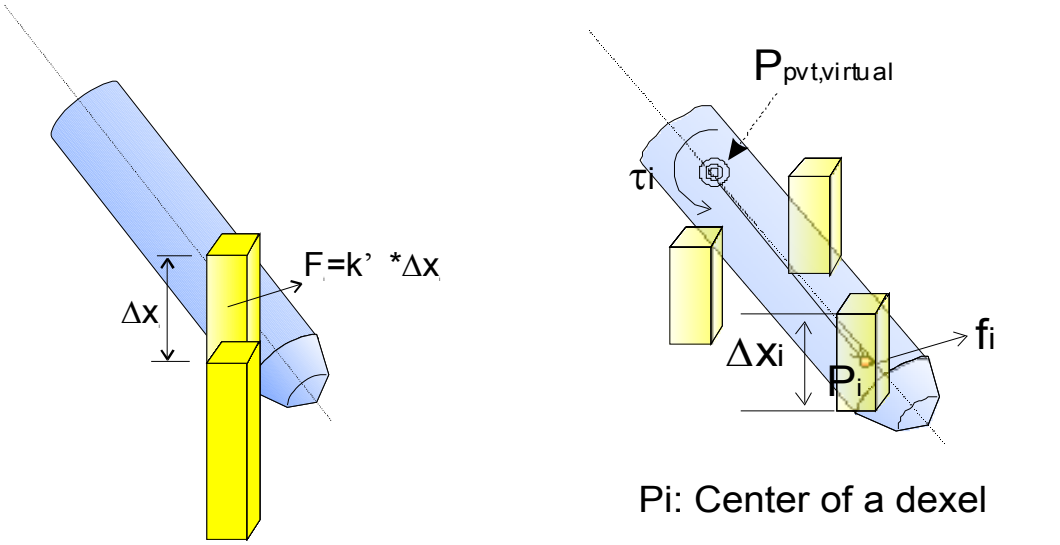
$$\vec{\tau}_i = f_{i,mag} \cdot (\vec{f}_{i,dir} \times (\overrightarrow{P_i P_{pvt,virtual}})) \tag{3.19}$$

where $\vec{f}_{i,dir}$ is the direction of a component force,

P_i is the center of the removed dexel,

$P_{pvt,virtual}$ is the virtual tool pivot point,

$\overrightarrow{P_i P_{pvt,virtual}}$ is the vector from P_i to $P_{pvt,virtual}$.



(a) Force is proportional to the removed Dexel height

(b) The calculation of torque based on the force and pivot point

Figure 3.18. Calculation of force and torque

The 'tool approaching direction' or force direction $\vec{f}_{i,dir}$ needs to be defined explicitly since it is a mass rod, instead of a mass point. As discussed above, there are four different kinds of tool movement situations. Accordingly, there are four different situations in locating the force exertion point and thus the torque calculation (see Figure 3.19):

I.) *Tool moves along the tool axis*: this is the simplest situation. There is no torque feedback in this case since the force is along the tool axis (see Figure 3.19(a)). The force direction is thus the same as the current tool orientation:

$$\vec{f}_{i,dir} = \begin{cases} Tool\ Orientation,\ if\ \ going\ down \\ -Tool\ Orientation, if\ \ going\ up \end{cases} \tag{3.20}$$

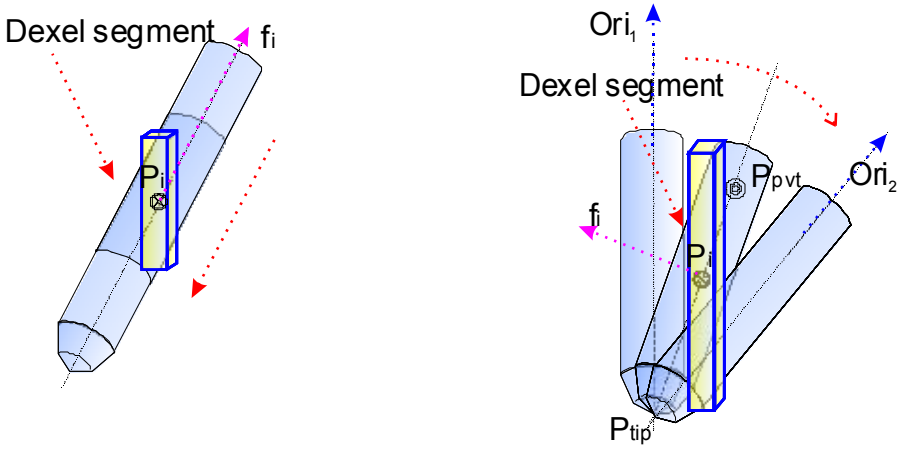
II.) *Tool rotates around an axis without translation*: there are both force and torque in this situation. Suppose the center of the removed dexel segment

is at point P_i, then the corresponding pivot point is on the straight line of P_i and the rotational center point (see Figure 3.19(b)). The force direction can thus be defined as perpendicular to the corresponding tool orientation.

$$\vec{f}_{i,dir} = Normalize((\overrightarrow{P_{i,tip}P_i}) \times (Ori_{i,1} \times Ori_{i,2})) \quad (3.21)$$

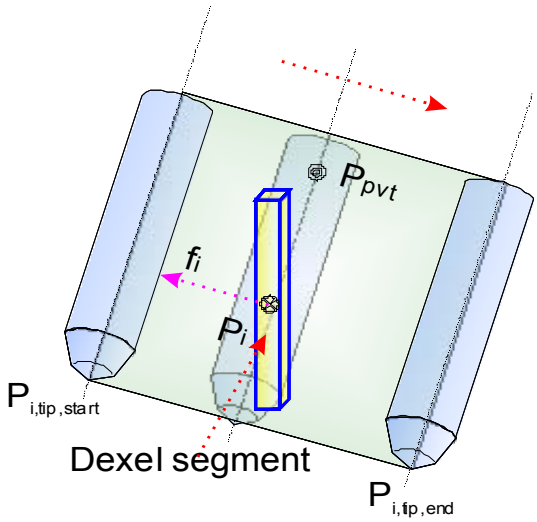
where $P_{i,tip}$ is the tool tip position,

$Ori_{i,1}$ and $Ori_{i,2}$ are the orientation of the rotation start and end tool, respectively.



(a) Tool moves up and down

(b) Tool rotates only



(c) Tool translates only (after discretization of tool motion)

Figure 3.19. Force-torque feedback calculation for 3 tool movement situations

III.) *Tool moves with translation only*: in this case, the 'tool approaching direction' is the vector from start tool tip position to end tool tip position:

$$\vec{f}_{i,dir} = Normalize(\overrightarrow{P_{i,tip,end}P_{i,tip,start}}) \tag{3.22}$$

where $P_{i,tip,start}$ and $P_{i,tip,end}$ are the tool tip positions of the start tool and end tool, respectively. There are both force and torque in this situation (see Figure 3.19(c)).

IV.) *Tool moves around with rotation and translation*: this is the most common case during virtual sculpting. Since the tool motion is discretized into parallel linear segments, in each linear segment, the tool is considered to be moving linearly. Hence, the 'tool approaching direction' in each discretized segment is also calculated by using Equation (3.22).

The force and torque calculated on each removed dexel are accumulated and fed back to a user through haptic interface. Assume there are totally *m* dexels removed in the current instance, the accumulated force and torque are calculated as follows (refer to Equations (3.18) and (3.19)):

$$\vec{f} = \sum_{i=1}^{m} \vec{f}_i\left(f_{i,mag}, \vec{f}_{i,dir}\right) = k'\left(\sum_{i=1}^{m} \Delta x_i \cdot \vec{f}_{i,dir}\right) \tag{3.23}$$

$$\vec{\tau} = \sum_{i=1}^{m} \vec{\tau}_i = k\sum_{i=1}^{m} (\Delta x_i \cdot (\vec{f}_{i,dir} \times (\overrightarrow{P_i P_{pvt}}))) \tag{3.24}$$

It is noted that in a comprehensive virtual environment, the tool may collide with some environmental models other than the Dexel model being sculpted. In this case, a relatively big force $\vec{F}_{collision}$ and torque $\vec{\tau}_{collision}$ are added to the final force-torque feedback to prevent the user from penetrating the some environmental models. Then the total force and torque can be calculated as:

$$\vec{f}_{total} = \vec{f} + \vec{F}_{collision} \tag{3.25}$$

$$\vec{\tau}_{total} = \vec{\tau} + \vec{\tau}_{collision} \tag{3.26}$$

In the current implementation, due to the limitation of computation capacity, the collisions with the environmental models are not included in the haptic virtual sculpting system. More discussion on collision force is presented in Chapter 4 [Zhu 2003d]. The calculated force and torque as in Equations (3.25) and (3.26) are distributed to the haptic arms and fed back to the user [Zhu 2003b].

3.4 Developing Haptic Interface for Virtual Sculpting

Figure 3.20 shows our lab setup of a lab-built 5-DOF (degree of freedom) pen-based electro-mechanical haptic device made by Suzuki, *Inc.*, Japan. It can detect 6-DOF of haptic probe movement and gives 5-DOF force and torque feedback. Figure 3.20 shows the lab setup with the haptic device located in the middle. In Figure 3.20, the right hand side is the constructed controller. Its left hand side is the dual 2.4G*Hz* CPU workstation with the implemented haptic rendering programs discussed in this chapter. The haptic device has a left-manipulator and a right one as shown in Figure 3.20.

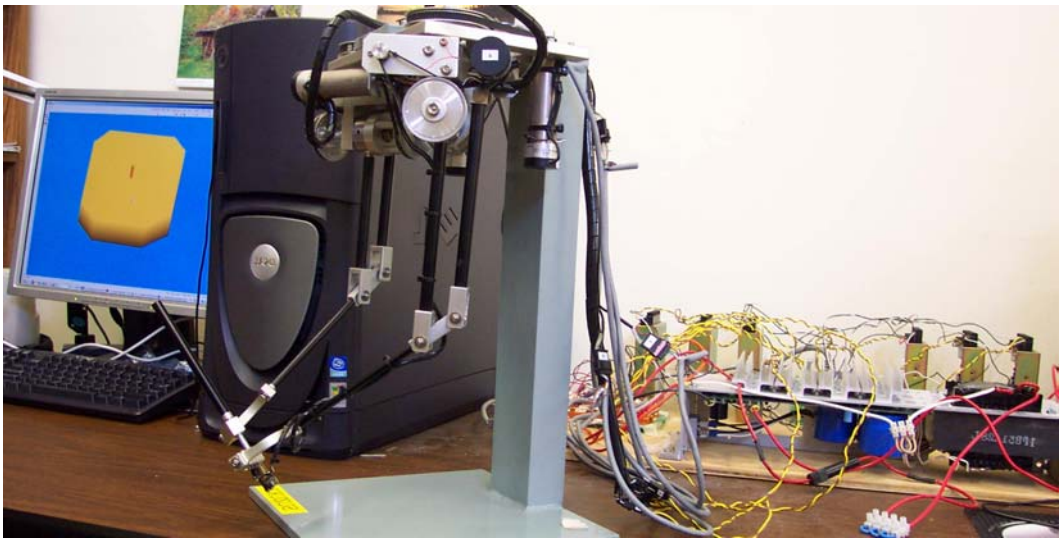


Figure 3.20. Lab setup of haptic device controller and 5-DOF haptic device, with a dual-CPU workstation

The force and torque calculated from Equations (3.23) and (3.24) pass through the hardware driver to drive the motors on the haptic device and to provide force feedback to the user. More details about the working principles of the haptic interface are presented in Chapter 2.

Due to the heavy computing task of Dexel-updating process and also considering the control stability of the haptic device, the haptic rendering software is composed of two cycles: *haptic cycle* and *graphic cycle* (see Figure 3.21). In a dual CPU computer, each cycle runs as a thread on one CPU. In the *graphic cycle* thread, the virtual stock material is updated according to the movement of the virtual cutter, which corresponds to the haptic probe. In this process, the Dexel volume model (stock material) is updated. At the same time, the force and torque are calculated according to the stock material removal, with Equations (3.23) and (3.24). The graphic rendering is refreshed every 50ms (20*Hz*). In the *haptic cycle*, one CPU keeps fetching the current force and torque calculated by the *graphic cycle* every 10ms (100*Hz*). In the meantime, every 1ms (1000*Hz*) this CPU interpolates between the previous force/torque and the current force/torque, and then output the interpolated force/torque. By doing so the output rate for the haptic device is sustained as 1000*Hz*, which is required to provide a continuous feeling of force/torque to the user.

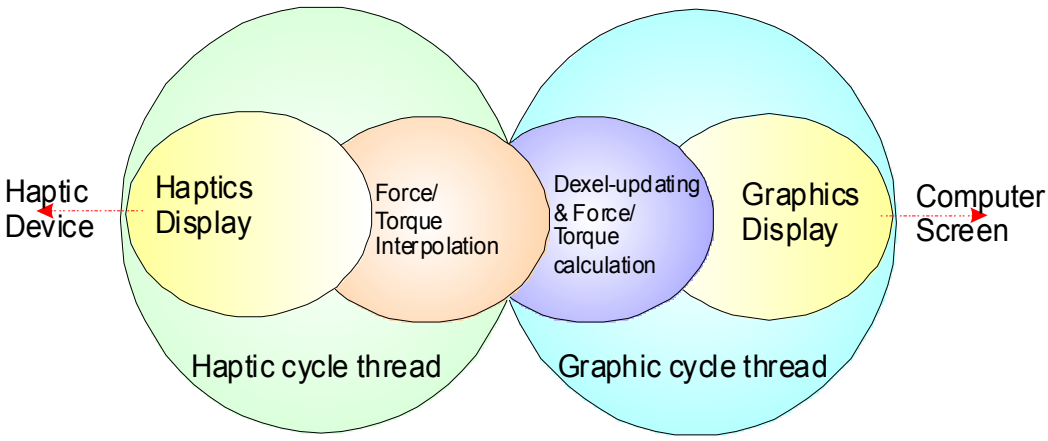


Figure 3.21. Haptic and computer interface system loops

An argument here is that the force felt by the user may not be the concurrent force (see Figure 3.22). This argument is true. However, this is acceptable since the graphic cycle refresh rate (20*Hz*) is much lower than the force/torque calculation rate (100*Hz*). Our later experiments also reported that no user can detect that the force/torque is not truly concurrent with the visual graphic display. On the other hand, if faster algorithms can be developed or the computer speed is much faster, it is best to sustain the haptic rendering as 1000*Hz* directly without interpolation.

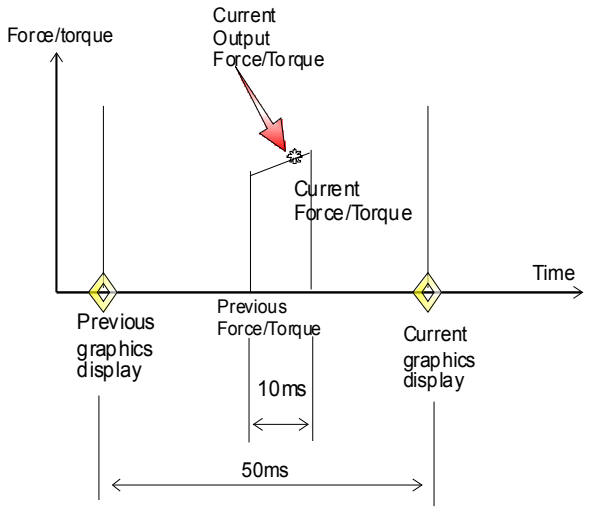


Figure 3.22. Force feedback and graphics display are not perfectly synchronous

3.5 Implementation and Examples

The above methodology has been implemented in the developed haptic sculpting system at North Carolina State University. As shown in Figure 3.20, the haptic system controller and the haptic rendering programming were implemented on a dual 2.4G*Hz* CPU workstation, using Visual C++ and OpenGL®. The computer also comes with 2G memory and a high-end video card. Since we constructed the haptic device controller and developed the haptic rendering programs from the hardware level, we own the greatest flexibility in designing our specific haptic applications.

Figure 3.23 shows an example operation process of the haptic sculpting system based on the proposed methodology. The user was moving the haptic probe of

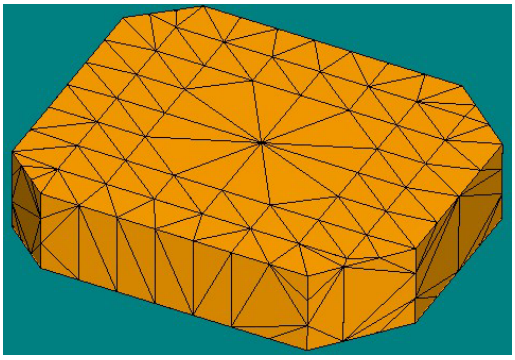
the haptic device. Concurrently the tool was displayed on the screen and sculpting the stock material. When the tool was carving the stock, the user can actually feel the force and torque. There are two forms of the output from the haptic sculpting system (both have been implemented as subsidiary functions of the system):

- The sculpted model can be output as STL polyhedral surface model. This turns out to be a difficult problem. The constructing of STL model and Dexel volume model is presented in Chapter 5 [Zhu 2003d].
- The tool motion can be recorded and output as NC (numerically-controlled) code command. In this case, the user's creative work can be recorded and played back at will.

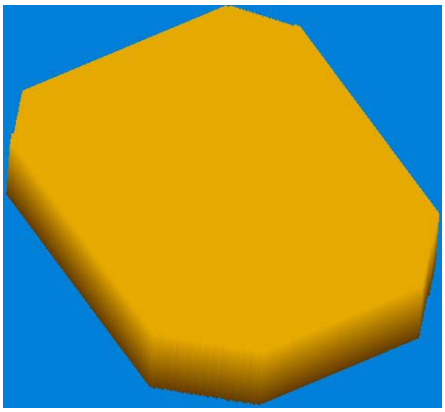
Figure 3.23. The experiment setup of the haptic sculpting system

Figure 3.24(a) shows the STL model of a stock material. It is converted into a Dexel model when it is opened in the haptic sculpting system (Figure 3.24(b)). A user defines a tool and then uses the tool to sculpt the stock material in Dexel model format. Figure 3.25 shows an example design of a bird modeled with the haptic sculpting tool. The sculpted stock material in Dexel model format is then exported as an STL model. The rendered view of the output STL model is shown in Figure

3.26(a). The output STL model is composed of about 87,000 triangles. Figure 3.26(b) shows some details near the head of the bird STL model.



(a) An STL model



(b) The STL model is converted into a Dexel model as the initial design

Figure 3.24. An STL model is converted into Dexel model

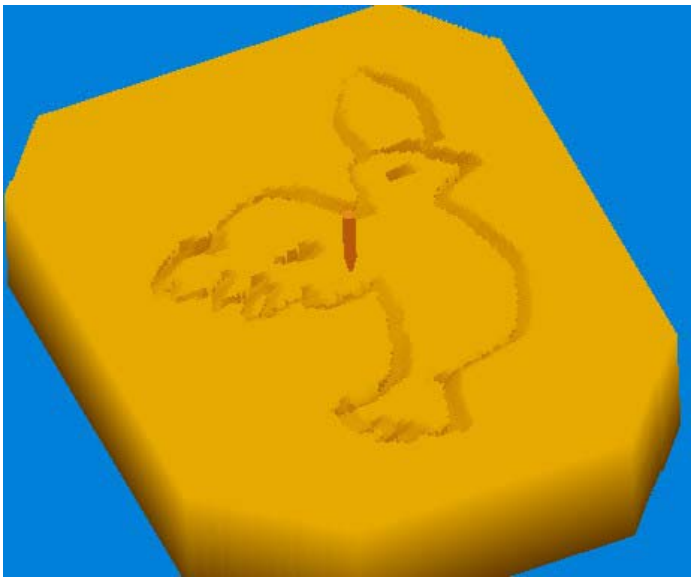
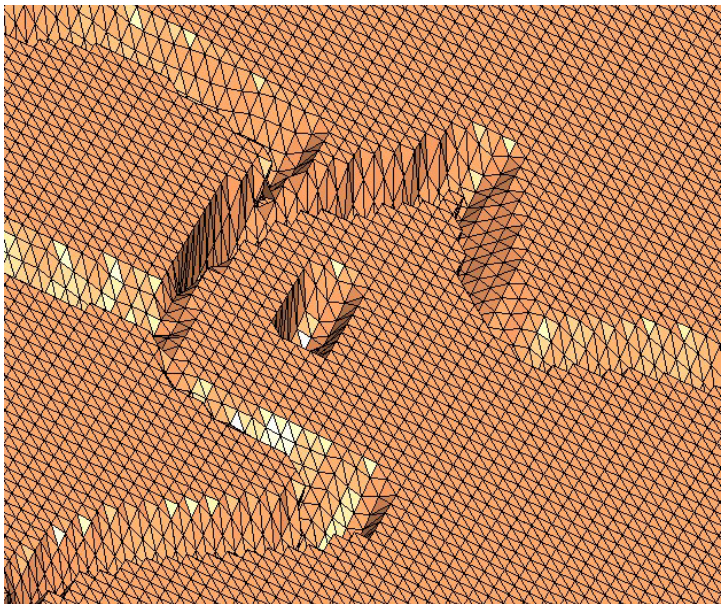


Figure 3.25. The bird Dexel model is being sculpted by using the haptic sculpting

(a) The rendered view of the output STL model from the haptic sculpting system

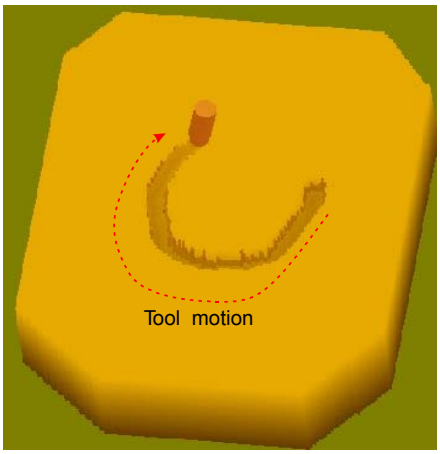


(b) Details near the head of the bird STL model from the system

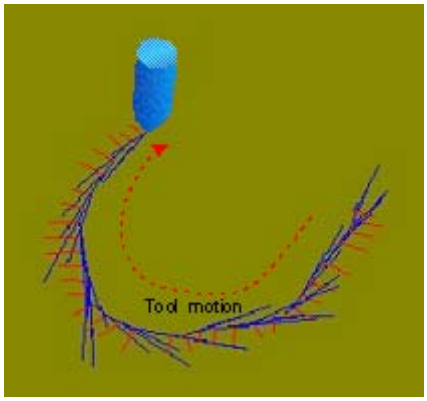
Figure 3.26. The sculpted bird Dexel model is exported as an STL model

Figure 3.27 shows an experiment on force and torque calculation. It was conducted during the development of the virtual sculpting system. A tool moves along a U shape path as in Figure 3.27(a). The force and torque calculated based on the tool motion is sampled and visualized as vectors in Figure 3.27(b). In a colorful printout,

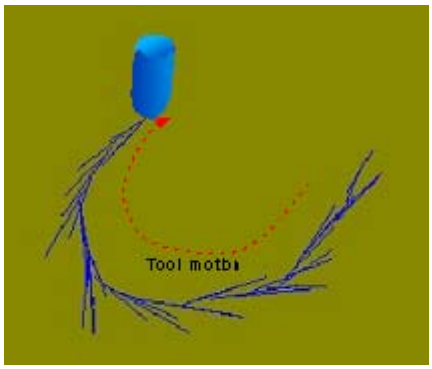
the force is in blue color and the torque is in red color. Considering a grayscale printout, the calculated force and torque are shown in Figures 3.27(c) and (d), respectively. It can be seen from Figure 3.27(c) that basically the force is following the tool path with changing magnitudes. In Figure 3.27(d), the torque is proportional to the force magnitude and its direction is determined by right-hand rule. More examples are presented in Chapter 6.



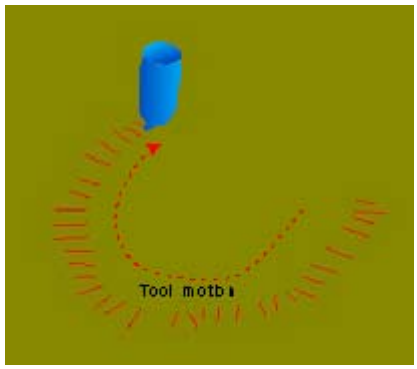
(a) A tool sculpting a stock material



(b) The sample force and toque shown together



(c) The sample calculated force, with direction and magnitude



(d) The sample calculated torque, with direction and magnitude (directions follow right-hand rule)

Figure 3.27. An experiment: visualization of force/torque feedback

3.6 Discussion

Compared to previous work in dexel-based haptic sculpting systems and commercially available haptic sculpting system, our methodology features:

i) *Different tool definitions*: commercially available free-form modeling only defines the tool tip as a simple sphere. The tool is defined as combination of 2D circular arcs and straight lines. Thus it can represent ball-end mill, flat-end mill and taper-end mill tool in the common NC machining.

ii) *Fast approximation of swept volume*: as a tool sweeps across the space, it generates a swept volume. As mentioned before, in haptic sculpting, the objective is to get fast haptic rendering by finding a trade-off between *Dexel-updating* precision and speed requirement. In this implementation, the swept volume of 5-axis tool motion (translation and rotation) is modeled and analyzed. The tool swept volume is used to update the stock material, which is represented with the Dexel model. The *Dexel-updating process* is decomposed into ray-parallelogram and ray-pipe intersection problems, *etc*. Analytical solutions are found for these problems.

iii) *5-DOF Dexel-based collision detection and force-torque feedback algorithm*: the current algorithm takes advantage of the Dexel-updating calculation and calculates the force-torque feedback by the removed Dexel segments. The method of force-torque feedback for 5-DOF output haptic device differs greatly from 3-DOF output haptic device.

iv) *Two forms of output*: there are two forms of output from the haptic sculpting system. The user's creative work process can be recorded as NC codes. In this way, the work can be played back at will. The created artwork can also be output as STL model, which can be further processed to be manufactured.

v) *Hardware integration*: The force and torque feedback calculation is tightly integrated with the hardware driver level. No commercial haptic library functions were utilized and haptic interface was constructed from the hardware level. In this case, although it was much harder to kick off at the beginning, it turns out that we could enjoy the greatest flexibility in designing and implementing the software algorithms.

3.7 Summary

In this chapter, an analytical methodology for haptic sculpting system of virtual prototyping is presented. Tool motion and analytical tool swept volume intersection are formulated for updating the virtual stock material represented in Dexel volume model format. A *Dexel Removal Rate* (DRR) analysis has been developed for haptic force-torque response calculation. Based on the analysis, a dexel-based collision detection method and a force-torque feedback algorithm are proposed for virtual prototyping. With the proposed methodology, a user can virtually sculpt a stock volume to get an intuitive design by using a developed 5-DOF (degree of freedom) force-torque haptic interface. From the haptic sculpting system, both tool path for the creative process and the sculpted model can be output. The output model of the virtual prototyping system may be processed by the machining planning algorithms developed in our lab to generate NC code for NC machining [Zhu 2003b]. The presented techniques can be used for virtual prototyping, CAD/CAM and machining planning.

Chapter 4

Machining Planning with Haptic Interface for Five-axis Pencil-cut

Machining Planning plays an important role in CAD/CAM systems. In this chapter, techniques of *5-axis pencil-cut* machining planning with a 5-DOF (Degree of Freedom) output haptic interface are presented. Five-axis tool path planning has attracted great attention in CAD/CAM and NC machining. For efficient machining of complex surfaces, *Pencil-cut* uses relatively smaller tools to remove the rest material at corners or highly curved regions that are inaccessible by larger tools. As a critical problem for 5-axis *pencil-cut* tool path planning, the tasks of tool orientation determination and tool collision avoidance are achieved with a developed 5-DOF haptic interface. Algorithms of haptic rendering are proposed for force and torque feedback calculation with haptic interface. A *Two-phase approach* to tool collision detection and local gouging elimination is proposed for pencil-cut of sculptured surfaces. A *Dexel-based method* is developed for global tool interference avoidance with other components in a machining environment. Detailed techniques of *haptic rendering* and *tool interference avoidance* are discussed for haptic-aided 5-axis *pencil-cut* tool path generation. Hardware and software implementation of the haptic *pencil-cut* system with practical examples are also presented in this chapter.

4.1 Introduction

Five-axis tool path planning has attracted great attention in Computer-aided Manufacturing (CAM) and NC machining. There are many issues of 5-axis machining to be addressed, among them the tool path generation and tool orientation control are two main issues. Great progress has been made in 5-axis tool path planning after years of research [Choi 1998, Jensen 2002, Jun 2003, Lee 1998]. In the CAM area, some

initial attempts and inspiring work concerning the utilization of haptic interface have been developed in MIT. They have produced some interesting results in 5-axis tool path generation. In their work, a quick collision detection method was proposed between a tool and a machining environment represented by point clouds [Ho 2001]. In their tool path generation application, they machined a part with constant Z-height machining method [Balasubra 2002].

This chapter presents the technique of employing haptic interface for 5-axis *pencil-cut* tool path planning. The remaining of this chapter is organized as follows. Section 4.2 introduces the problem of *pencil-cut* machining operation. Section 4.3 describes the methodology of the haptic-aided 5-axis *pencil-cut* tool path generation in details. Section 4.4 discusses the software and hardware issues in implementing the haptic interface system. Practical examples and experiment results are presented in Section 4.5, followed by concluding remarks and future work.

4.2 Problems of Pencil-cut Machining of Sculptured Surfaces

In sculptured surface machining, there are several stages of machining including roughing, semi-finishing, finishing and clean-up [Choi 1998]. The result of roughing removes the bulk material outside the design surface. After semi-finishing and finishing, the shape of the machined stock is almost similar to the designed part surface. Clean-up machining removes those rest material left at the sharp corners or edges. *Pencil-cut* is used for clean-up machining (Figure 4.1). *Pencil-cut* in 3-axis machining has been researched for years and this function is provided in some commercial software for parametric surface. In our earlier work presented in [Ren 2002], algorithms for 3-axis *pencil-cut* of polyhedral models have been developed for sculptured surface machining. To the best of our knowledge, research work on 5-axis *pencil-cut* has not been reported.

Five-axis *pencil-cut* has the obvious advantage over 3-axis *pencil-cut*, due to its better accessibility with the additional rotation axes. For 3-axis *pencil-cut*, the tool

orientation is fixed as the Z-axis, as shown in Figure 4.1. Five-axis *pencil-cut* is especially useful for cleaning rest materials in complex surfaces. As can be seen from Figure 4.2, the curved corner region can be accessed by 5-axis machining, but not accessible by a 3-axis machining. In our earlier work presented in [Ren 2002], a method has been developed to identify the clean-up regions on complex part surface. For 5-axis *pencil-cut*, the difficulty for tool path planning lies mostly in the tool orientation determination.

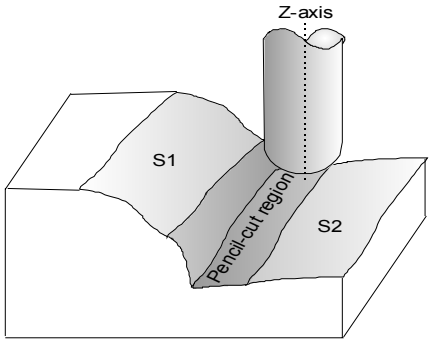


Figure 4.1. Three-axis pencil-cut

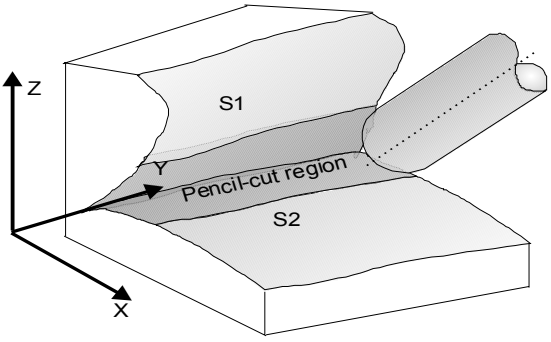
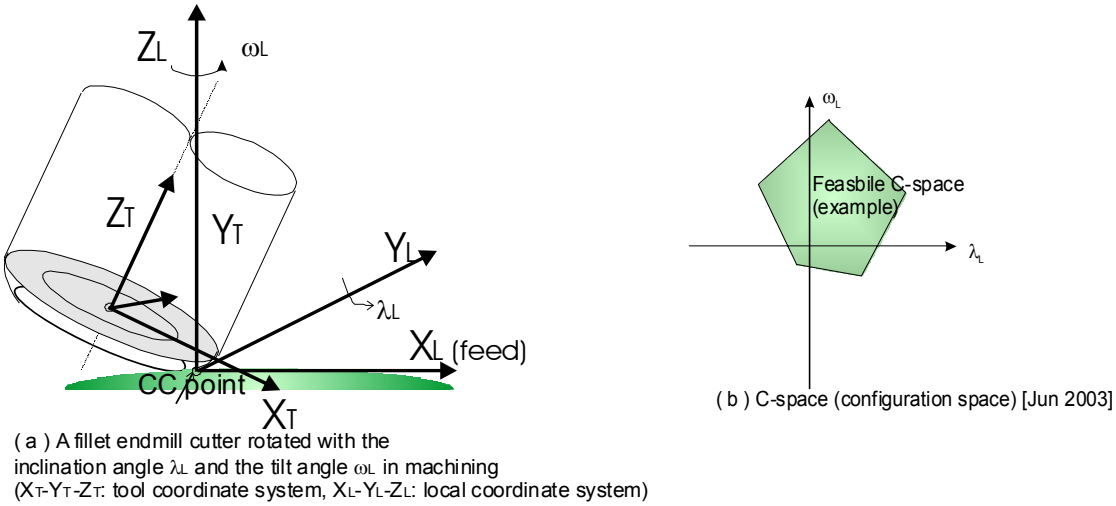


Figure 4.2. Five-axis pencil-cut

In 5-axis machining, a cutting tool can be oriented by rotating around one axis with an inclination angle λ_L and another axis with a tilt angle ω_L, as shown in Figure 4.3. To facilitate searching for a feasible tool orientation, usually a local coordinate system is set up as follows: X_L axis is the instantaneous feed direction (cutting direction); Z_L axis is the surface normal direction at the current CC (cutter contact) point; Y_L axis is determined by the right-hand rule (see Figure 4.3(a)) [Lee 1998]. Then the tool orientations are searched based on this local coordinate system. To find a good tool orientation control in 5-axis machining, a *C-space* (configuration-space) is formed with the inclination angle λ_L and the tilt angle ω_L, as shown in Figure 4.3(b) [Choi 1998]. As discussed in [Jun 2003], a machining *C-space MCS* of a given CC point CC_i on a designed part surface *PS* machined with a cutter Ψ can be defined as follows (also see Figure 4.3(b)):

$$MCS(\Psi, PS, \{CC_i\})_{i=0,\ldots,n} = \begin{bmatrix} \lambda_L \\ \omega_L \end{bmatrix} \tag{4.1}$$

Past research on the tool orientation mainly adjusts the inclination angle λ_L, due to the theoretically infinite combinations of inclination angle λ_L and tilt angle ω_L. Most of the research only considers the *local* optimal tool orientation, which is based on analysis of the part surface property and cutting conditions near the CC point.



(a) A fillet endmill cutter rotated with the inclination angle λ_L and the tilt angle ω_L in machining (X_T-Y_T-Z_T: tool coordinate system, X_L-Y_L-Z_L: local coordinate system)

(b) C-space (configuration space) [Jun 2003]

Figure 4.3. Coordinate systems in 5-axis machining and C-Space definition

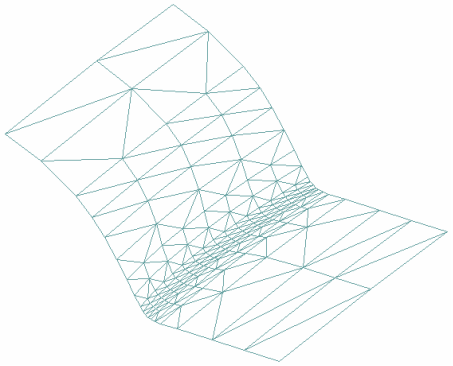
When seeking for the *globally optimized* tool orientation in 5-axis machining, we have to deal with expensive computation and searching concerning the interference between the whole tool assembly with the stock. The reason is that we have to find the feasible (not necessarily the optimal) tool orientation by trial and error. There are several attempts in circumventing this exhaustive search [Lee 1995]. However, there are also some limitations in the results, due to the different assumptions made on the designed part surface or the cutting tool [Jun 2003].

In this chapter, we developed a haptic system to help determine and select the tool orientations for 5-axis *pencil-cut* machining in a complex machining environment [Zhu 2003a and Zhu 2003b]. Details of developed the haptic system are discussed in the following sections.

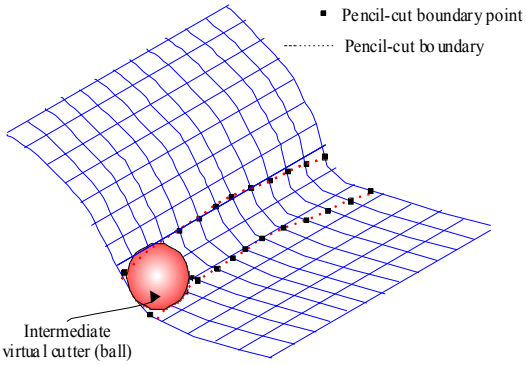
4.3 Proposed Haptic System for 5-axis Pencil-cut Machining

4.3.1 Identification of pencil-cut region and tool path generation

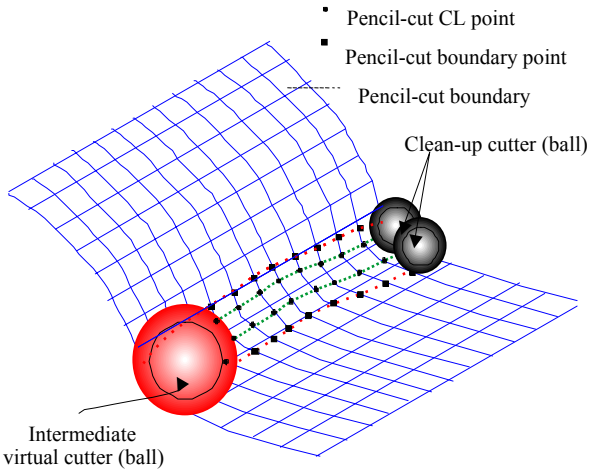
The first step for *pencil-cut* machining is to identify where the *pencil-cut* operation should be executed. In this chapter, a designed surface is represented in STL (stereo lithography) format, which is composed of a collection of triangles and basically is a kind of polyhedral model (see Figure 4.4(a)). The use of STL (stereo lithography) format for representing a CAD model has been widely accepted in industry for some time [Koc 2000]. As shown in Figure 4.4, a *Rolling-ball method* developed in our earlier work presented in [Lee 2000, Ren 2002] was revised to extract the *pencil-cut* regions of sculptured surfaces. The Rolling-ball method works in the way described as follows.



(a) an illustrative model for pencil-cut



(b) Identifying pencil-cut region



(c) Tracing pencil-cut tool paths with two smaller balls

Figure 4.4. Rolling ball method for identifying pencil-cut regions

of sculptured surfaces [Ren 2002]

Figure 4.4(a) shows an example polyhedral model. A *CC-net*, which is a net of CC (cutter contact) points, is cast on the polyhedral model, as shown in Figure 4.4(b). A ball with an appropriate radius, which is usually set as the radius of finishing cutter, rolls along this CC-net to identify the *pencil-cut* boundary points [Lee 2000]. Two smaller balls, whose radii are usually set as half of the radius of the previous big reference ball, are rolling along a route parallel to the *pencil-cut* boundary, as shown in Figure 4.4(c). In this way, the *pencil-cut* regions can be identified and the parallel *pencil-cut* tool paths can be traced for each *pencil-cut* region, as detailed in [Ren 2002].

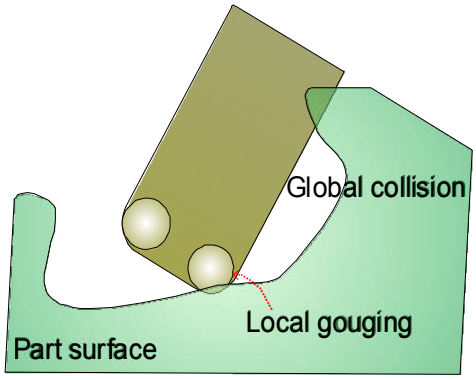


Figure 4.5. Global collision and local gouging in 5-axis NC machining

For 5-axis *pencil-cut*, the tool orientations are to be determined, by considering both the local surface property and the surrounding machining environments. These problems of *global tool collision* and *local tool gouging* in 5-axis machining, as shown in Figure 4.5, still need to be solved for the determination of 5-axis tool orientation control. More detailed discussion of solving these problems will be discussed later in Section 4.4.

4.3.2 Developed haptic interface for 5-axis pencil-cut tool path planning

After the *pencil-cut* regions are identified, the feasible 5-axis cutter trajectory and tool orientation are determined by using haptic interface. The working principle and the development of the haptic device have been presented in Chapters 2 and 3. As

shown earlier in Figure 3.20, the haptic device has a left-manipulator and a right one. Both of them working together feed back the force and torque to the user. Figure 4.8 shows the corresponding force and torque configuration in the virtual tool assembly with the physical haptic device probe. Assume that from the virtual reality calculation, it is desired that the left manipulator needs to generate force $\vec{f}_L$ and the right manipulator needs to generate force $\vec{f}_R$, then the necessary torques $\vec{\tau}_L$ and $\vec{\tau}_R$ generated from the motors for haptic rendering can be calculated by the *Jacobian* Matrices (J_L^T and J_R^T), as they have the following relation (see Figure 4.8 and also refer to Equations (2.5) and (2.6)):

$$\vec{\tau}_L = J_L^T \vec{f}_L \tag{4.2}$$

$$\vec{\tau}_R = J_R^T \vec{f}_R \tag{4.3}$$

The derivation of the *Jacobian Matrix* is dependent on the link structure of a specific haptic device, as detailed in Chapter 2 [Craig 1989]. Equations (4.2) and (4.3) will be used later in this chapter. The details of force $\vec{f}$ and torque $\vec{\tau}$ feedback calculation for the haptic interface will be discussed in Section 4.4, as part of haptic rendering.

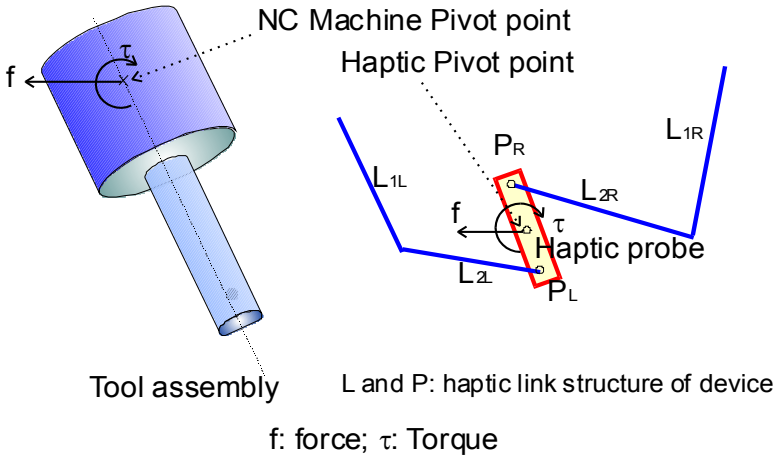


Figure 4.6. The transformation between the tool assembly and the physical haptic system link structure

In the virtual tool assembly, the pivot point is defined as rotation center of a NC machine spindle head, as shown in Figure 4.6. In the physical haptic device, the pivot point is the middle point between the two ends of the left and right manipulators (Figure 4.6). The pivot point is used for calculating the torque required in the virtual world.

4.3.3 Determining the feasible 5-axis tool orientation using haptic interface

Generally in 5-axis NC machining, the interference between the workpiece system and the tool assembly can be divided into two types (see Figure 4.5):

- *Global collision* between the workpiece, fixtures and the non-cutting portion of the tool assembly, e.g. tool holder and tool shank;
- *Local gouging* between the designed part surface and cutting portion of the tool.

The user needs to eliminate both of them when choosing a feasible 5-axis tool orientation. During the haptic interaction process, the user can easily orient the tool assembly to avoid *global collision*, as one can feel the force $\vec{f}$ and torque $\vec{\tau}$ feedback from the haptic system in the case of tool collision. On the other hand, *local gouging* is relatively difficult to be completely eliminated. During the haptic interaction process, the virtual tool touches the surfaces, which means there must be at least a slight interference between the tool and the designed surface. Hence *global collision* is removed during the haptic interaction. Post-processing for the tool path is invoked to eliminate *local gouging*. Details on how interferences are detected and corrected will be discussed in Section 4.4.

Figure 4.7 shows an example surface designed for demonstrating *pencil-cut* tool path planning. Two parallel *pencil-cut* tool paths were identified along the edges using *Rolling-ball method* mentioned earlier. During the haptic interaction process, the nearest CC point to the current virtual tool tip is found and highlighted with its normal vector, as shown in Figure 4.7. To speed up the computation and to reduce

interaction load of the haptic system, it is proposed that only the tool orientations of those critical CC points need to be specified and the other CC points' tool orientation are to be defined by interpolation. This is analogous to animation creation, where key animation frames are defined and interim animation frames are automatically created by interpolation. During the interaction process, if a certain CC point is picked up as a key point, the user moves the virtual tool to that CC point, orients the virtual tool along an appropriate orientation, based on what she/he sees and the force and torque she/he feels at the haptic probe, and presses a hot key to record the current tool orientation for this CC point.

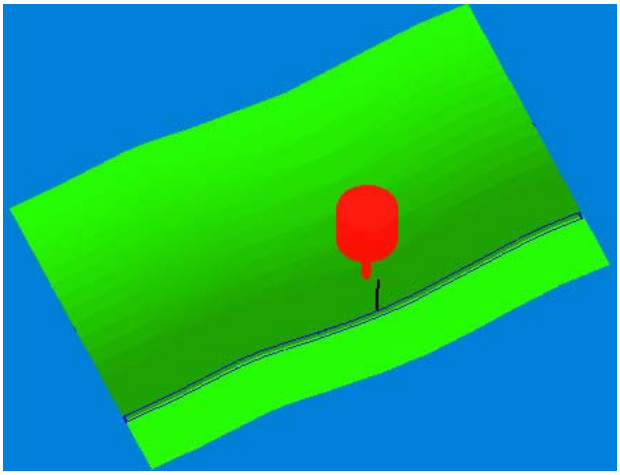


Figure 4.7. Pencil-cut tool path (two blue parallel paths) identified on an example surface, with the nearest CC point checked out and highlighted with its normal vector

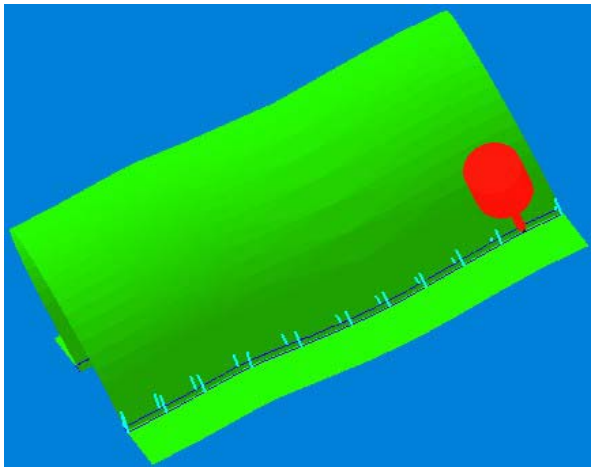
A complex machining environment can be composed of workpiece, jigs and fixtures, *etc*. The interference between the tool and all these components should be considered and thus affects the tool orientation selected. Figure 4.8(a) shows the critical tool orientations specified by a user via haptic probe. The selection of the critical tool orientations can be done based on the graphical and haptic feedback of force $\vec{f}$ and torque $\vec{\tau}$. These critical tool orientations usually locate at where surface features changes. Figure 4.8(b) shows the interpolated tool orientations between the critical tool orientations. If the user is unsatisfied with the current results, she/he can choose to re-specify the critical tool orientation or adjust the interpolated

tool orientations directly, until all the orientations are satisfying. Assume two adjacent critical tool orientations are $\overrightarrow{O_1}$ and $\overrightarrow{O_2}$ and there are n CC points between them, the i-th tool orientation on of the i-th interim CC point is interpolated as (see Figure 9):

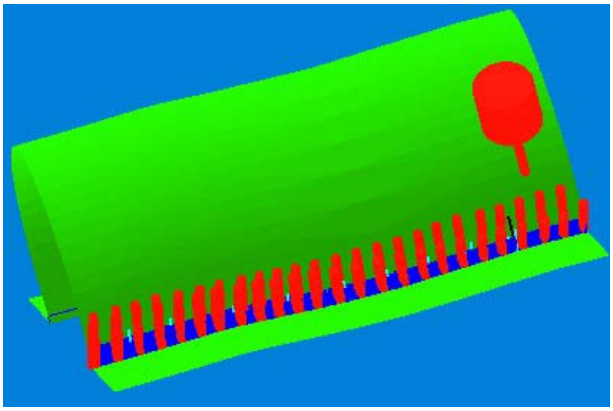
$$\overrightarrow{O_i} = \frac{i \cdot (\overrightarrow{O_2} - \overrightarrow{O_1})}{n+1} + \overrightarrow{O_1} \tag{4.4}$$

$$\overrightarrow{O_{i,unit}} = Normalize(\overrightarrow{O_i}) \tag{4.5}$$

This will be illustrated more clearly in the examples presented later in Section 4.5. The calculation of interference detection and force feedback is presented in Section 4.4.



(a) Critical tool orientations (cyan vectors in light color) specified on the example surface



(b) Interpolated tool orientations (blue vectors in dark color) and some sample tool orientations on the example surface

Figure 4.8. Tool orientations generated for the *pencil-cut* tool paths

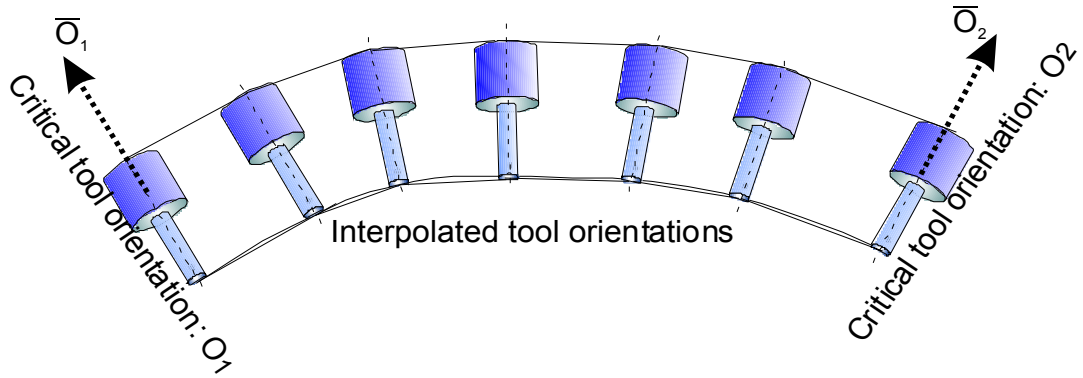


Figure 4.9. Tool orientation interpolations from two critical tool orientations

4.3.4 Post-processing and eliminating local gouging in tool path generation

After the tool orientations are selected, the tool path information is post-processed to eliminate local gouging. In the post-processing, the tool orientation for each CC point is kept intact and the tool assembly is checked for local gouging. The tool is lifted up to a position along the current tool orientation to eliminate any local gouging. Figure 4.10 illustrates this process. After the automated post-processing, the tool path can be generated by computing CL (cutter location) points $\overrightarrow{CL_i}$ from the CC points $\overrightarrow{CC_i}$ with the following formula for filled-end mill (see Figure 4.11):

$$\overrightarrow{CL_i} = \overrightarrow{CC_i} + r_{tc} * \overrightarrow{n_i} + \frac{e \cdot ((\overrightarrow{n_i} \times \overrightarrow{O_i}) \times \overrightarrow{O_i})}{\left|((\overrightarrow{n_i} \times \overrightarrow{O_i}) \times \overrightarrow{O_i})\right|} + r_{tc} * \overrightarrow{O_i} \qquad (4.6)$$

where r_{tc} is the minor radius of a fillet-end mill,

e is the major radius of a fillet-end mill,

$\overrightarrow{n_i}$ is the normal vector associated with the current CC point,

$\overrightarrow{O_i}$ is the tool orientation for the current CC point.

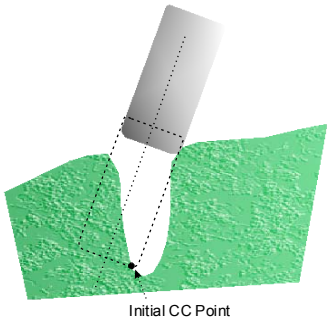


Figure 4.10. Retracting cutter to avoid local gouging

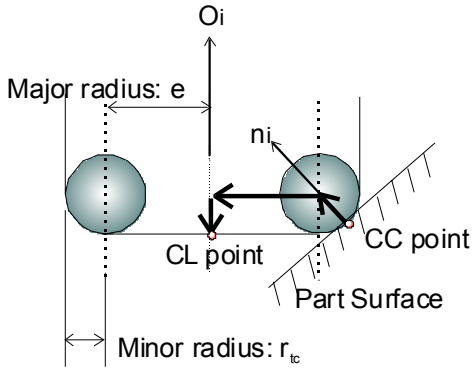


Figure 4.11. Computing CL points from CC Points

The CL data is further processed to generate NC code to drive the NC machine. In this chapter, the generated NC code is simulated on commercial NC simulation and verification software. Simulation results for a practical part surface are presented in Section 4.5.

4.4 Haptic Rendering Programming for 5-axis Pencil-cut

Haptic rendering programming first checks the interference/collision (including both *global collision* and *local gouging*) between a moving tool assembly and the static product surface in a complex machining environment. If any interference is detected, the program continues to calculate the extent of interference and to calculate the required force and torque in the virtual world. The desired force and torque are converted to haptic manipulator force $\vec{f}_L$ and $\vec{f}_R$, as indicated earlier in Equations (4.2) and (4.3), so that they can be used to drive the haptic device and fed back to the user.

The collision detection speed is critical for haptic applications. The update rate is estimated to be 1,000 Hz in order to maintain a sustained feeling of force feedback [Biggs 2002]. In every 1 millisecond, the computer needs to detect the coordinates of the haptic probe, transform the coordinate values from the physical device coordinate system to the graphics environment, detect the collisions between the virtual objects and calculate the force feedback. Object / object collision detection methods are required in the current application. In order to achieve high update rate for complex environments, computing techniques like the spatial decomposition tree [Klosowski 1998, McNeely 1999] and the OBB (object oriented bounding-box) tree [Gottschalk 1996] are adopted in this chapter.

In the current application, both the designed part surface and its fixtures are considered to constitute the surrounding machining environment. The designed part surfaces are assumed to be freeform surfaces with complex geometry. The fixtures, on the contrary, are most likely prismatic and more regular in shape. In this chapter, a *Two-phase* approach for tool collision detection and local gouging elimination is proposed for haptic *pencil-cut* of sculptured surfaces. A *Dexel-based* volume modeling is used for global tool interference avoidance with other components, like fixtures, of a machining environment. Both approaches are detailed in the following sections.

4.4.1 Two-phase rendering approach to collision detection and force response for sculptured surfaces

In this chapter, a *Two-phase* rendering approach is proposed for the haptic pencil-cut interface and the rendering programming, as shown in Figure 4.12. Our method is described as follows (also see Figures 4.12 and 4.13). In the *Two-phase* approach, for convenience of the first phase, the triangular polygons in STL models are organized into OBB (object oriented bounding-box) trees, as shown in the 'First phase' of Figure 4.13.

In the first phase (see Figures 4.12 and 4.13), the *possible collision triangles* (PCT) are extracted by checking the interference between the OBB-trees of STL surface models and the simple OBB-tree of the tool assembly. In our approach, the tool definition is a combination of implicit surfaces. To improve the computing efficiency, we try to simplify the definition of the tool while trying to accommodate a more complex part surface. Triangulated STL model is adopted in this chapter because the computation and geometric processing of STL model are relatively faster. The next step is to find out how we can quickly detect the collision and calculate the force feedback. STL surface model's OBB-tree is static during the interaction process. During the movement of haptic probe, the tool assembly's OBB-tree is updated (Figure 4.13). There is little overhead in updating the tool assembly's OBB-tree.

In the second phase (see Figures 4.12 and 4.13), to improve the efficiency of collision detection, the triangles in STL model are discretized into sampling point clouds at the beginning of the haptic application, as shown in the 'Second phase' in Figure 4.13. Each point is associated with a normal direction, which is set as the normal direction of the triangle it belongs to. Adaptive point cloud density is used for different features of the tool assembly. In the second phase, the points corresponding to the PCT (*possible collision triangles*) are checked against the tool's implicit surface for collision test, as shown in Figure 4.12. If a point is inside the tool assembly, we find the nearest tool surface point to the collision point and then calculate the distance between the collision point to the corresponding tool surface point. The force magnitude is proportional to this distance. Assuming this distance is Δx_i and k is a pre-defined coefficient, the responding force magnitude $f_{i,mag}$ is calculated as follows:

$$f_{i,mag} = k \cdot \Delta x_i \tag{4.7}$$

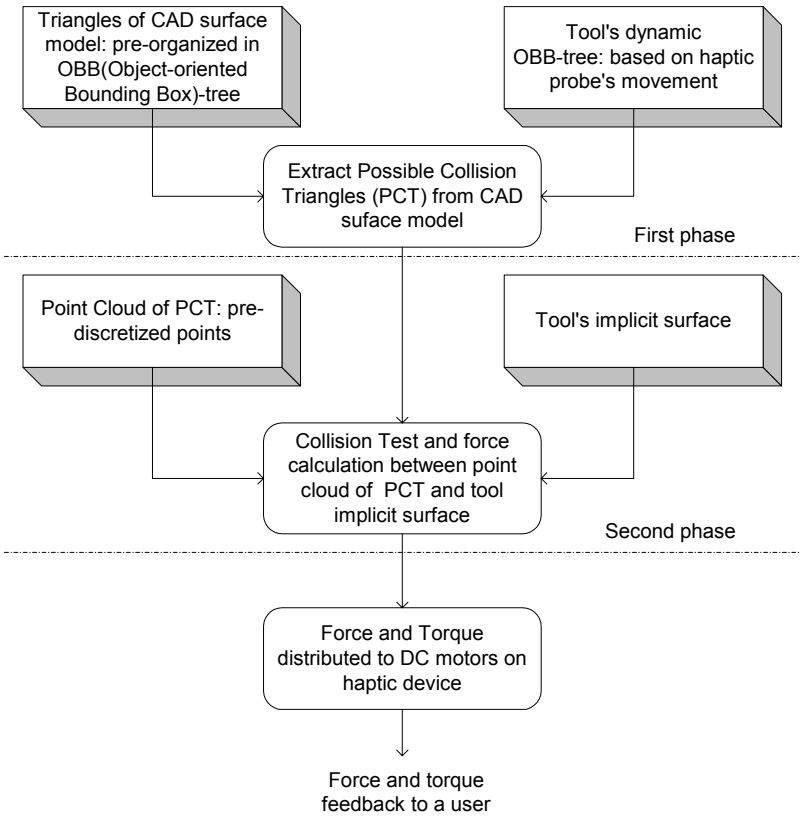


Figure 4.12. Two-phase approach for haptic rendering

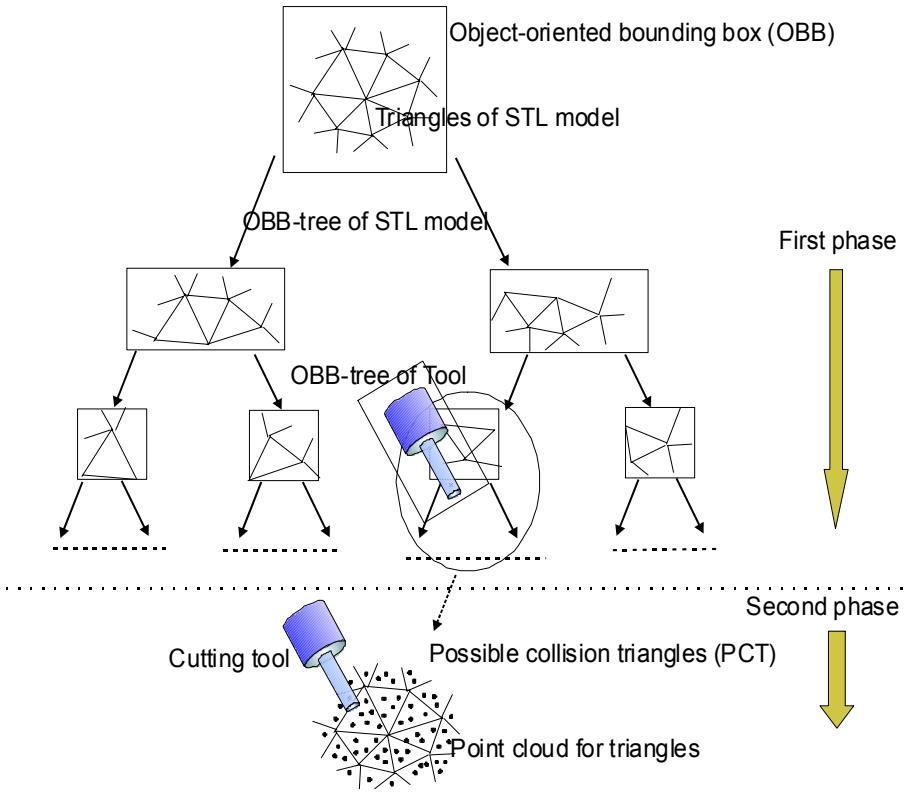


Figure 4.13. Two-phase haptic rendering: collision detection and force response

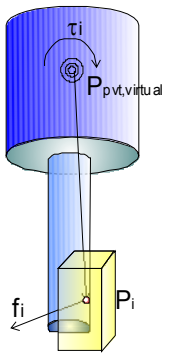


Figure 4.14. Calculation of torque feedback, given a collision point P_i

The unit force direction $\vec{f}_{i,dir}$ has already been associated with the collision point as mentioned earlier in the process of generating point cloud. A component force $\vec{f}_i(f_{i,mag}, \vec{f}_{i,dir})$ is determined by its magnitude $f_{i,mag}$ and direction $\vec{f}_{i,dir}$. Assume the virtual pivot point's location is $P_{pvt,virtual}$, the collision point's location is P_i, and the force magnitude of $f_{i,mag}$ is calculated by Equation (4.7), the torque τ_i induced by this collision point is calculated as follows (Figure 4.14):

$$\vec{\tau}_i = f_{i,mag} \cdot (\vec{f}_{i,dir} \times (\overrightarrow{P_i P_{pvt,virtual}})) \qquad (4.8)$$

Equations (4.7) and (4.8) will be used for accumulation of total force $\vec{f}$ and torque $\vec{\tau}$ fed back to the user. This will be presented in Section 4.4.3. The advantages of the proposed *Two-phase* approach include:

i) The OBB-tree used for the haptic application environment can speed up the computation efficiency significantly, especially when the sculptured surfaces become complex;

ii) The densities of the point clouds are adaptive for different features of a tool;

iii) The cutting tool's shape can be complex and a generalized fillet-end mill [Chiou 1999] is used as the general representation for different endmills of ball-end mills, flat-end mills, taper-end mills and fillet-end mills;

iv) The force and torque feedback calculation (will be discussed in next section) is tightly integrated with the hardware driver level.

4.4.2 Dexel-based global tool interference avoidance with the machining environment

Dexel volume model is a popular format which is widely used in NC simulation [Van Hook 1986]. It represents a CAD model with many columns packed together (see Figure 4.15). Each column, as the basic element, is called a *Dexel*. Basically it has a 2-D grid and on each grid point, there is a linked list of Dexels. Usually a high density of Dexel grid is needed to represent a freeform surface. For most of the fixtures in the machining environment, however, the shapes are usually prismatic and regular and thus the density doesn't have to be high. Comparatively the haptic rendering method introduced in Section 4.4.1 is computationally intensive because the point cloud density could be high. On the other hand, for prismatic objects, Dexel models may be a less expensive solution for haptic rendering. Hence STL models of fixtures can be converted to Dexel models in a virtual machining environment. A virtual tool can be used to check against fixtures, which are represented with Dexel models, for collision detection and force-torque feedback calculation.

Figure 4.16 shows a virtual tool interferes with a fixture represented with a Dexel model. The interfered Dexels can be identified. The force magnitude $f_{Dj,mag}$ is proportional to the length of intersection Δz_{Dj} between the tool and the Dexels, as shown in Figure 4.17, shown as follows:

$$f_{Dj,mag} = k_D \cdot \left(\Delta Z_{Dj} \cdot Dx_{area}\right) = \left(k_D \cdot Dx_{area}\right) \cdot \Delta x_i = k'_D \cdot \Delta x_i \qquad (4.9)$$

where Δz_{Dj} is the intersection Dexel length,

Dx_{area} is the cross section of the Dexel elements,

k_D is the pre-defined cross-section dependent force coefficient,

k'_D is the cross-section independent force coefficient.

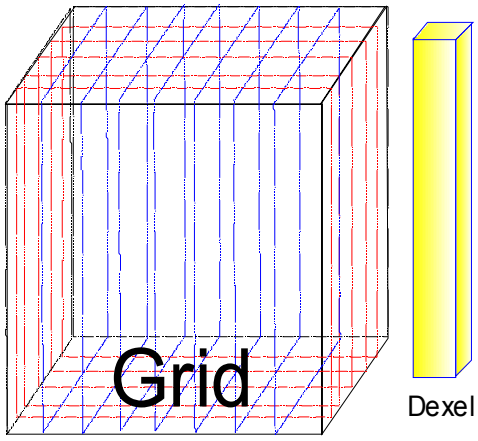


Figure 4.15. Dexel model format, used for haptic rendering of the other components in the machining environment

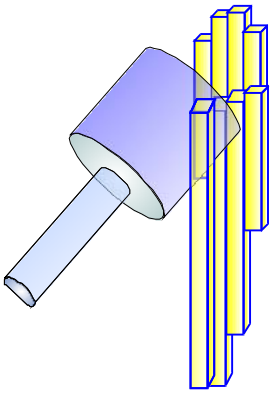


Figure 4.16. A virtual tool interferes with a Dexel model

Now the force direction $\vec{f}_{Dj,dir}$ needs to be defined. As shown in Figure 4.17, the whole cutter assembly is considered to be composed of two portions: holder and cutter. The geometric center of holder and cutter portions can be defined respectively as P_{holder_center} and P_{cutter_center}. The tool assembly may be intersected with Dexel model on holder or cutter portion. Assume that the geometric enter of the intersected dexel element is $P_{D,mid}$, the force direction $\vec{f}_{Dj,dir}$ is defined as follows (Figure 4.17):

$$\vec{f}_{Dj,dir} = \begin{cases} \overrightarrow{P_{D,mid}P_{holder\_center}}, & for\ holder\ portion \\ \overrightarrow{P_{D,mid}P_{cutter\_center}}, & for\ cutter\ portion \end{cases} \tag{4.10}$$

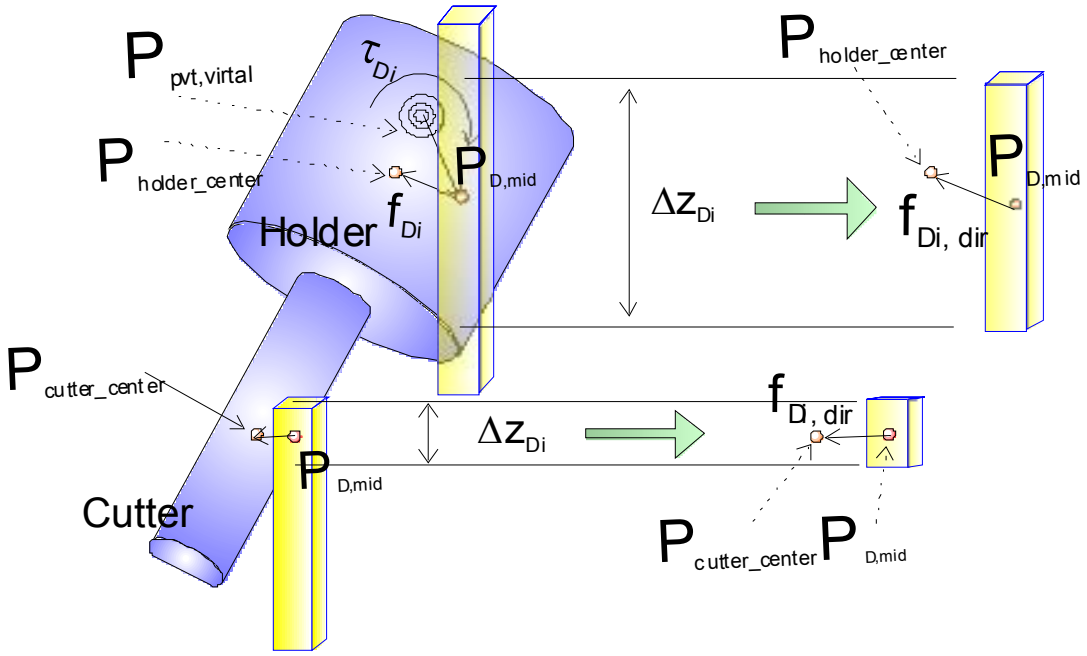


Figure 4.17. Calculation of torque feedback, given a collision point P_{Di} in Dexel model

A component force $\vec{f}_{Dj}(f_{Dj,mag}, \vec{f}_{Dj,dir})$ is determined by its magnitude $f_{Dj,mag}$ and direction $\vec{f}_{Dj,dir}$. Assume the virtual pivot point's location is $P_{pvt,virtual}$, the center location of the intersected Dexel is P_{Dj}, and the force magnitude of $f_{Dj,mag}$ is calculated with Equation (4.9), the torque τ_{Dj} induced by this collision point is calculated as follows (see Figure 4.17):

$$\vec{\tau}_{Dj} = f_{Dj,mag} \cdot (\vec{f}_{Dj,dir} \times (\overrightarrow{P_{Dj}P_{pvt,virtual}})) \tag{4.11}$$

4.4.3 Force and torque feedback distribution to the haptic device hardware

Two kinds of force need to be distinguished in the haptic rendering:

1) *Tracing force*: as the tool moves on the surface and traces along the tool path, the force feedback is tracing force (Equations (4.7) and (4.8)).
2) *Collision force*: if the holder portion of the tool collides with part surface (Equations (4.7) and (4.8)), or if the tool collides with the fixtures in the

machining environment (Equations (4.9) and (4.10)), the force feedback is collision force. The collision force is relatively strong compared to the tracing force.

Assume there are totally m collision points and q Dexel intersections in the current instance, from Equations (4.7)~(4.11), the force $\vec{f}$ and torque $\vec{\tau}$ feedback via the haptic probe can be calculated as follows:

$$\vec{f} = \sum_{i=1}^{m} \vec{f}_i + \sum_{j=1}^{q} \vec{f}_{Dj} = k \sum_{i=1}^{m} (\Delta x_i \cdot \vec{f}_{i,dir}) + k'_D \sum_{j=1}^{q} (\Delta z_{Dj} \cdot \vec{f}_{Dj,dir}) \tag{4.12}$$

$$\vec{\tau} = \sum_{i=1}^{m} \vec{\tau}_i + \sum_{j=1}^{q} \vec{\tau}_j = k \sum_{i=1}^{m} (\Delta x_i \cdot (\vec{f}_{i,dir} \times (\overrightarrow{P_i P_{pvt}}))) + k'_D \sum_{j=1}^{q} (\Delta z_{Dj} \cdot (\vec{f}_{Dj,dir} \times (\overrightarrow{P_{Dj} P_{pvt}}))) \tag{4.13}$$

Force $\vec{f}$ is distributed to two manipulators as follows (see Figure 4.20),

$$\vec{f}_{Lf} = f / 2 \tag{4.14}$$

$$\vec{f}_{Rf} = f / 2 \tag{4.15}$$

Assume that in Figures 4.6 and 4.18, the vector from left haptic manipulator end P_L to right manipulator end P_R is $\vec{r}_{LR}$, the torque is distributed to two manipulators as follows (see Figure 4.20):

$$\vec{f}_{L\tau} = \frac{\vec{\tau}}{|\vec{r}_{LR}|} \cdot \frac{\vec{\tau} \times \vec{r}_{LR}}{|\vec{\tau} \times \vec{r}_{LR}|} = -\vec{f}_{R\tau} \tag{4.16}$$

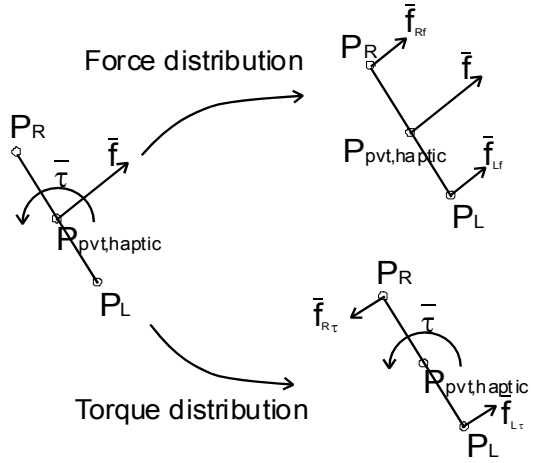


Figure 4.18. Force and torque distribution to two haptic manipulators

Then the desired forces on the left and right manipulators are calculated as follows:

$$\vec{f}_L = \vec{f}_{Lf} + \vec{f}_{L\tau} \tag{4.17}$$

$$\vec{f}_R = \vec{f}_{Rf} + \vec{f}_{R\tau} \tag{4.18}$$

The desired forces $\vec{f}_L$ and $\vec{f}_R$ are then substituted into Equations (4.2) and (4.3) to get the corresponding DC motor torques $\vec{\tau}_L$ and $\vec{\tau}_R$. The rotation of the DC motors applies force on the user's hand through the haptic device structure.

4.5 Implementation and Examples

The proposed techniques and the haptic hardware have been developed and implemented in our lab at North Carolina State University (see Figure 3.20). Based on the developed haptic controller system, the haptic rendering programs and the software driver have been implemented for this haptic system. The haptic controller and haptic rendering program were implemented on a dual 2.4GHz CPU workstation, with Visual C++ and OpenGL®. Interaction scheme is designed for the haptic application.

Figure 4.19 shows the overview of a user's operation of the haptic 5-axis *pencil-cut* system. Figure 4.20 shows an example part of a computer mouse model in STL format. As shown in Figure 4.20, two parallel *pencil-cut* tool paths are identified along the sharp edges by using the *Rolling ball method* developed in our earlier work in [Lee 2000, Ren 2002]. Figure 4.21 shows some sample 5-axis tool orientations for machining of the pencil-cut critical regions of the example part. Figure 4.22 (a) shows the generated 5-axis pencil-cut tool paths for machining the *pencil-cut* regions on the example part surface. The tool holders are temporarily hidden in Figure 4.22 to show the tool orientations more clearly. Without considering the surrounding machining environment, 5-axis tool paths could easily collide with the adjacent object unintentionally. It can be seen from Figure 4.22(b)

that the generated 5-axis tool paths are actually colliding with the fixtures nearby during 5-axis tool motions. As demonstrated in Figure 4.22(b), these tools actually penetrate into the fixtures. By taking the whole machining environments into account, the haptic system actually responds and enables the user feel the collision force feedback. By using the haptic interface, the user is able to correct the tool orientations. When one feels the force and torque feedback and sees the movement of tool, she/he can orient the tool to avoid or correct the global collision with ease.

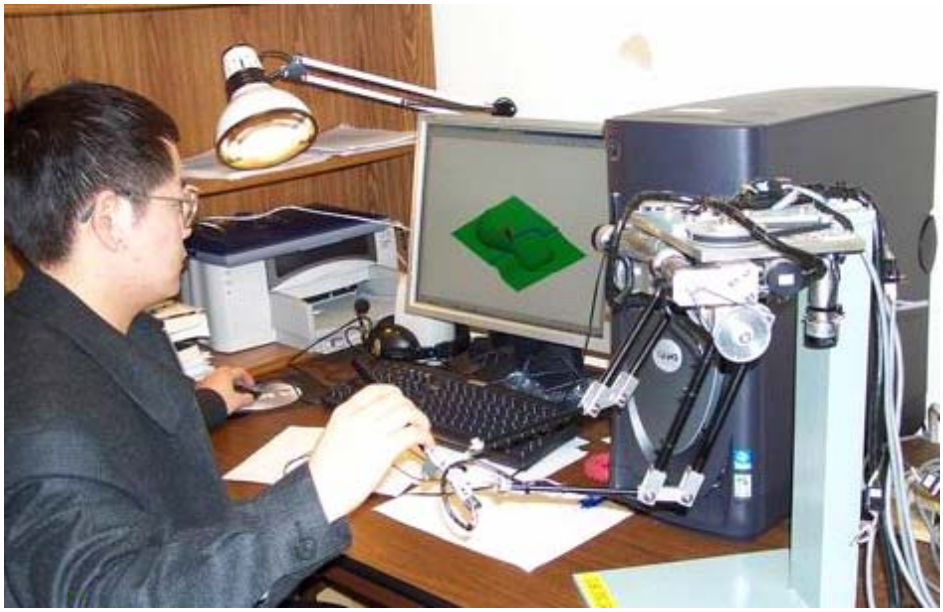
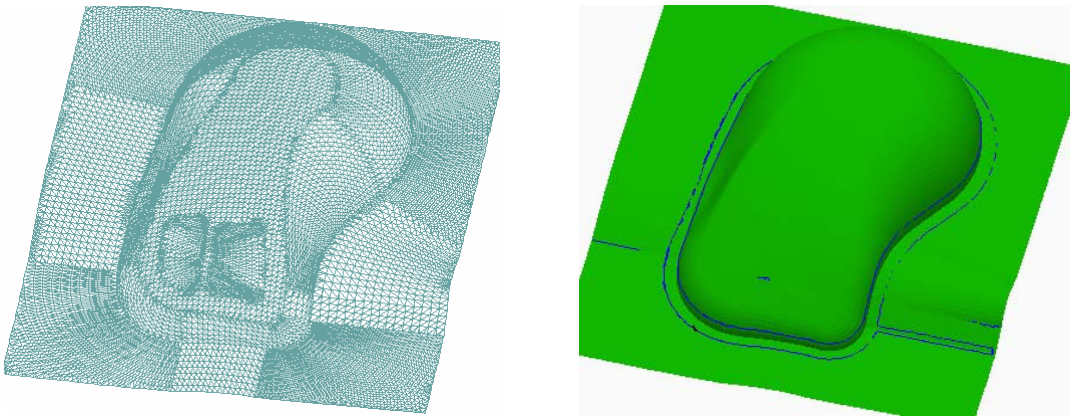


Figure 4.19. A view of operation process of the 5-axis tool path planning



(a) A mouse STL model (b) Pencil-cut tool path on mouse STL model

Figure 4.20. Pencil-cut tool path (dark blue lines) identified on a mouse surface model

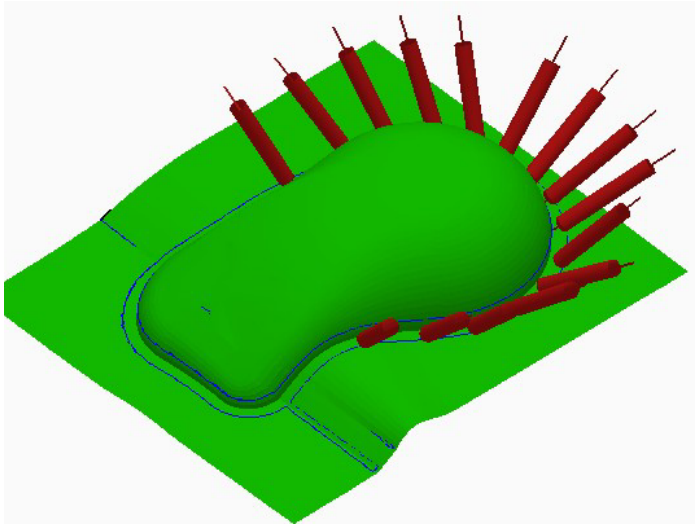
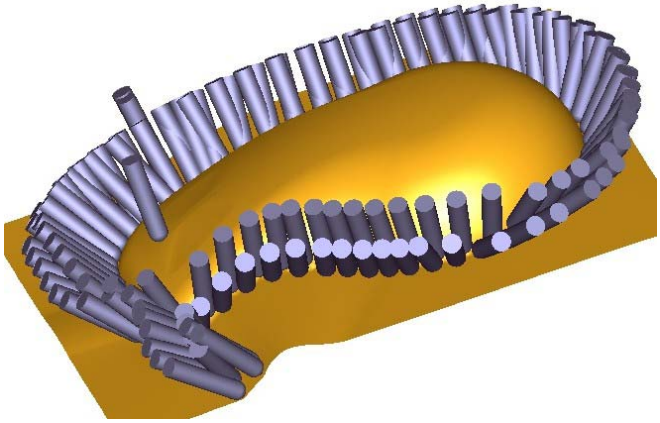
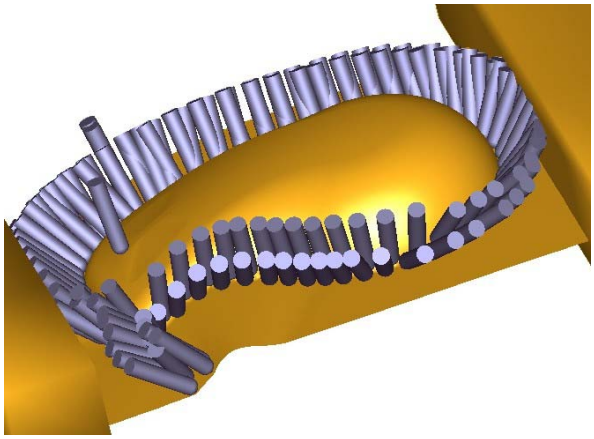


Figure 4.21. Tool orientations selected for pencil-cut of the example CAD model



(a) Tool orientations selected for pencil-cut tool on the mouse model, when there is no fixtures



(b) The selected tool orientations in (a) actually have global collision with fixtures: the tool penetrates into the fixtures virtually

Figure 4.22. Tool orientations selected without considering the machining environment may collide with the fixtures

Figure 4.23 shows the corrected 5-axis *pencil-cut* tool orientations generated with the developed haptic system. Figure 4.24 shows the re-computation of the corrected tool orientations for 5-axis pencil-cut after the global collisions have been eliminated by using the haptic system. Local gouging is further eliminated by the automated post-processing of the tool paths, as described earlier in Section 4.3. In Figure 4.25, the vectors in cyan color (light color in grayscale print-out) represent those critical tool orientations specified by the user during the haptic interaction. In Figure 4.25, the vectors in blue color (dark color in grayscale print-out) represent those interpolated tool orientations. Figure 4.25 shows the generated tool paths and the corrected tool orientations are also rendered in a view where sample tools are displayed with both the example part surface and the surrounding fixture. In Figure 4.25, it shows how the tool orientations are changing along the 5-axis *pencil-cut* tool paths without colliding with the surrounding workholders.

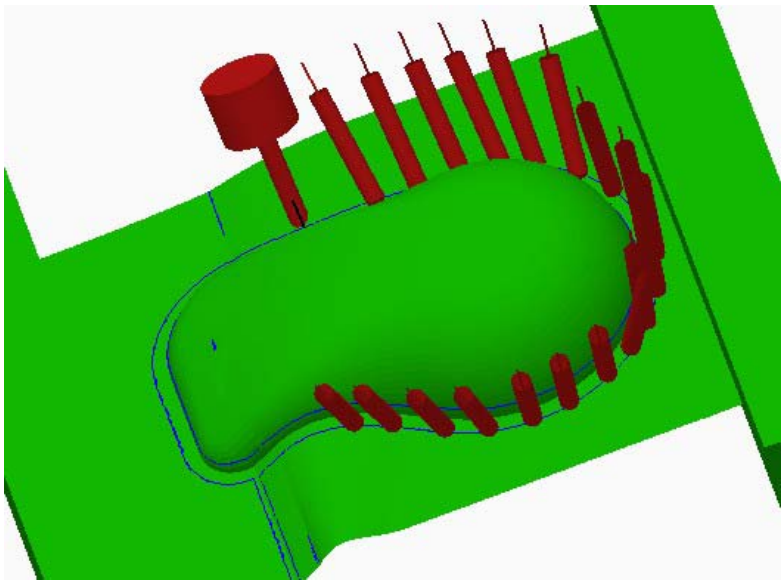


Figure 4.23. Corrected tool orientation selection
in a complex machining environment

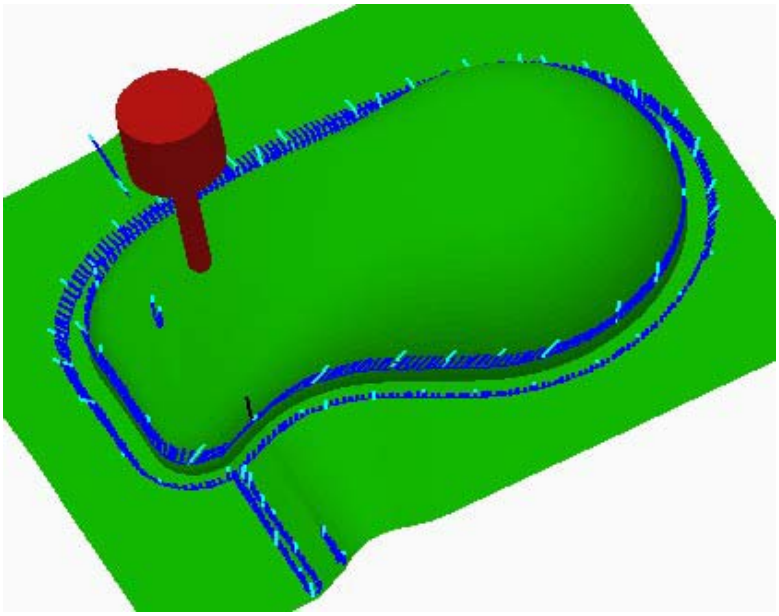


Figure 4.24. Re-computing pencil-cut tool orientations
after correction of tool collisions

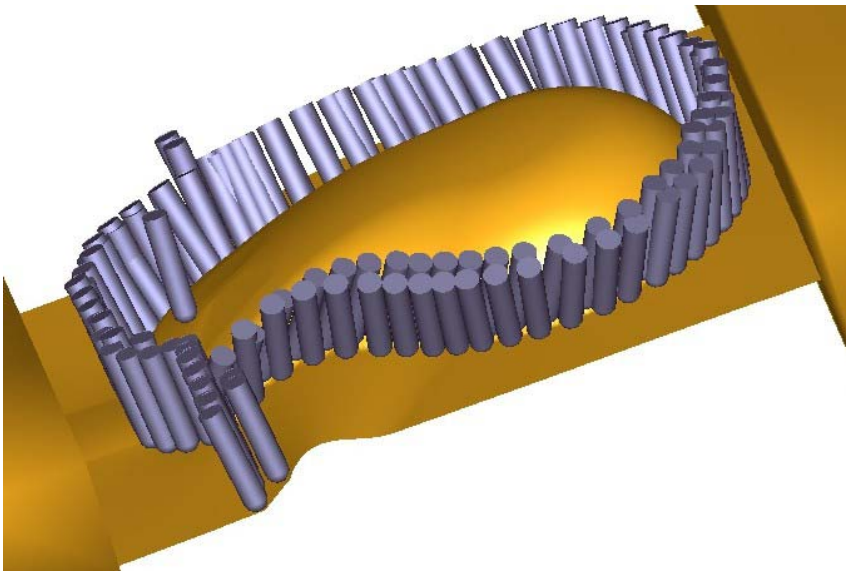
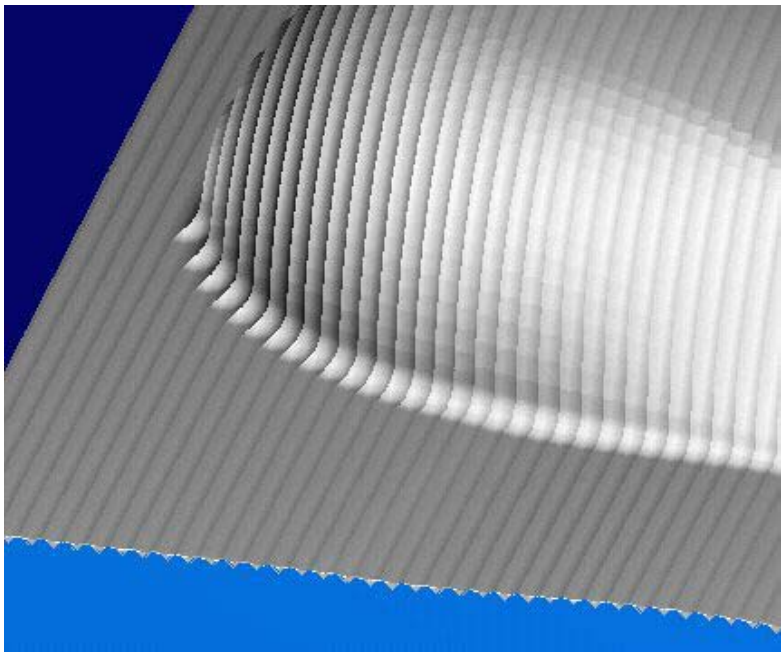


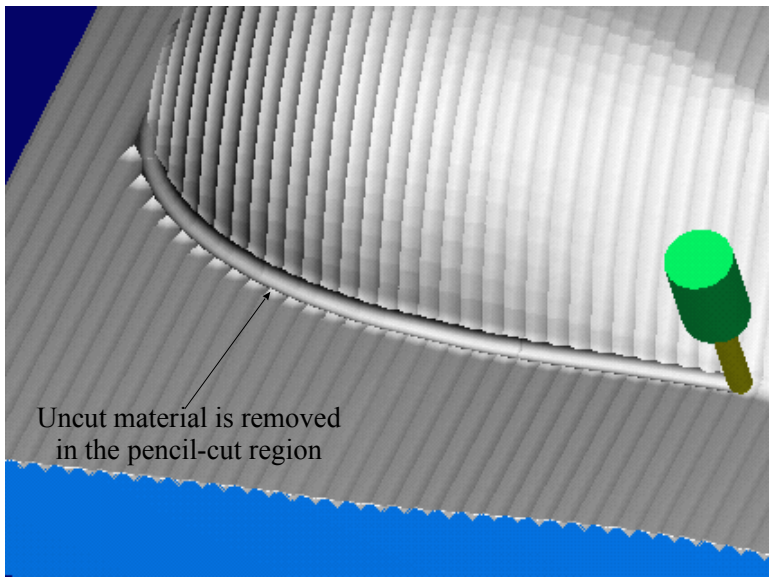
Figure 4.25. Corrected tool orientation selection
in a complex machining environment

The final outputs of the haptic system are the 5-axis NC tool paths. The generated NC code is simulated by using commercial NC simulation software in our lab. Figure 4.26(a) shows the example part surface after finishing. Figure 4.26(b)

shows the example part surface after *pencil-cut* operation has been performed with the generated NC code. The rest material near the shared sharp edges has been cleared by the pencil-cut tool path, as shown in Figure 4.26(b). From these examples, it can be seen that the resultant 5-axis *pencil-cut* tool path generated by the haptic-computer interface is quite satisfying. More examples are presented in Chapter 6.



(a) Mouse model after finishing without pencil-cut



(b) Mouse model during pencil-cut

Figure 4.26. Effects of 5-axis pencil-cut on the example mouse surface model

4.6 Summary

In this chapter, *haptic rendering* approaches have been developed for *5-axis pencil-cut machining planning* in a complex machining working environment. A *Two-phase* rendering approach has been proposed for *haptic 5-axis pencil-cut* of complex sculptured surfaces. *Dexel-based* modeling is used for global tool interference avoidance with other components of a machining environment. The presented techniques enable the haptic device be utilized to help determine feasible 5-axis tool orientations in a complex machining environment. Although the haptic-computer system described here mainly deals with *5-axis pencil-cut machining* operations, it is actually also used for other kinds of complex surface machining strategies [Zhu 2003b]. The experiments presented in this chapter show that the involvement of haptic system in 5-axis tool path planning is promising. The presented technique can be used for CAD/CAM, machining planning and virtual prototyping.

Chapter 5

Constructing Polyhedral Models from Dexel Volume Models for Haptic Sculpting

This chapter presents a *conversion marching algorithm* of constructing polyhedral models from Dexel volume models for haptic virtual sculpting. *Dexel volume model* is used as the in-process model during interactive modification in a virtual sculpting system. The stock material represented in Dexel volume model is sculpted into a designed model using a developed haptic sculpting system. The sculpted Dexel volume model can be converted to polyhedral model in STL format, which can be input to and processed by available CAM (Computer-aided Manufacturing) or RP (Rapid Prototyping) systems. The conversion turns out to be an interesting and challenging problem. This chapter presents a marching method to solve the problem. It is composed of three sub-algorithms: *roof and floor covering*, *wall-building* and *hole-filling* algorithms. The presented technique can be used in virtual sculpting, CAD/CAM and NC (numerically-controlled) machining.

5.1 Introduction

Polyhedral model is used extensively in CAD/CAM, Reverse Engineering (RE), Rapid Prototyping (RP), Finite Element Analysis (FEA), *etc*. The most commonly used polyhedral model is STL model, which represents an object with triangular facets [Koc 2000]. Figure 5.1 shows an example STL model of a computer mouse with 23,400 triangular facets. STL model files can be either in text (ASCII) or binary format for CAD/CAM or RP systems. Besides some header information, a STL model file is composed of a list of triangles [Choi 2003]. Each triangle facet in the STL model has the coordinates of its three vertices and also the normal vector. Since a triangle is usually adjacent to several other triangles, many vertices are duplicated.

Figure 5.1 shows the rendered view of an example computer mouse STL model. It can be seen that the example model is composed of many small triangular facets. STL model is a kind of surface model which records the triangulated shell of the object. On the other hand, volume models, such as *Voxel* or *Dexel* models, represent objects with small building blocks. Figure 5.2(a) shows the modeling method of volume model: *Voxel* model. Voxel model represents an object by packed unit building blocks. It is used extensively in *Volume Graphics* [Kaufman 1993]. Figure 5.2(b) shows the modeling method of Dexel volume model. Dexel model represents the objects with many packed columns. It is widely used in NC simulation and verification processes [Van Hook 1986, Huang 1994]. It takes less number of building blocks compared to Voxel model. Figure 5.3(a) shows the same example mouse model in Dexel format with a low resolution for easier identification of the Dexel elements. For comparison, Figure 5.3(b) uses a high resolution Dexel model to represent the mouse model and thus turns out to be smoother.

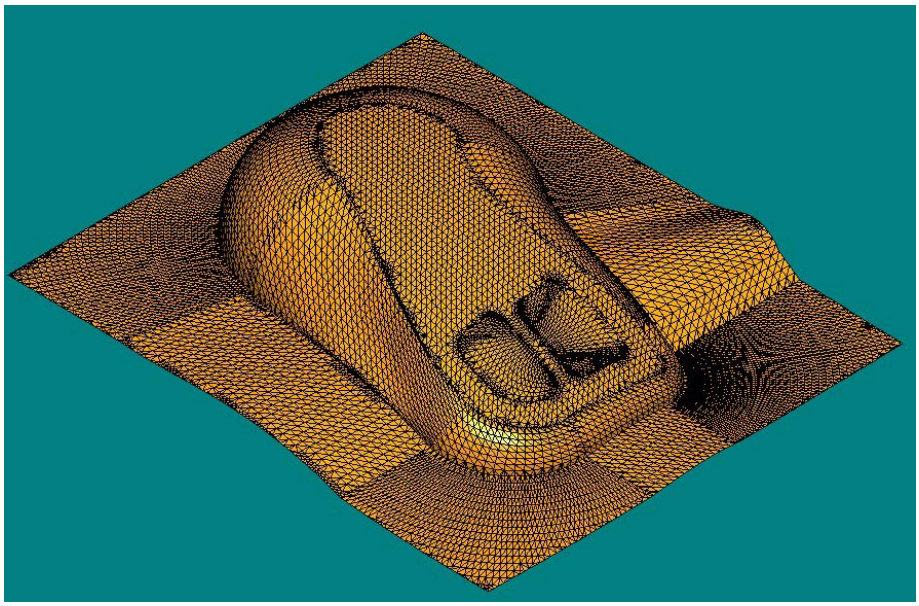
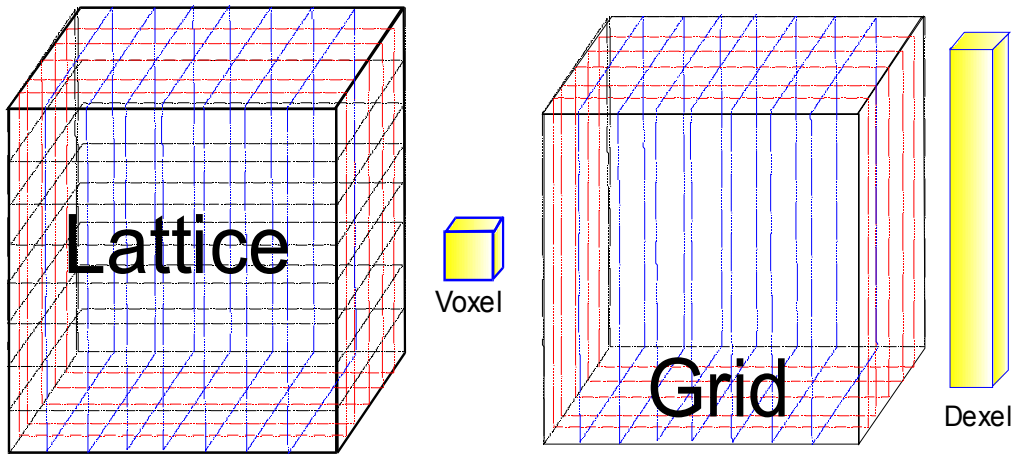


Figure 5.1. A rendered example STL model with about 23,400 triangular facets

In our research work presented in Chapter 3 [Zhu 2003a], a virtual sculpting system with a haptic interface has been developed. In the developed virtual sculpting system, a user can use a virtual tool to cut a virtual material block to get a creative and

intuitive design. As the cut object, the virtual material block can be represented in Dexel volume model format. During the virtual sculpting process, the swept volume of the virtual tool is subtracted from the virtual material block, if the tool and the block interfere with each other [Zhu 2003c]. The raw stock is represented as a Dexel volume model for virtual sculpting process to facilitate the calculation of its modification. After it is carved in the virtual sculpting system, the sculpted Dexel volume model needs to be transformed into STL model for CAM or RP (Rapid Prototyping) processing [Choi 2003, Zhu 2003d].

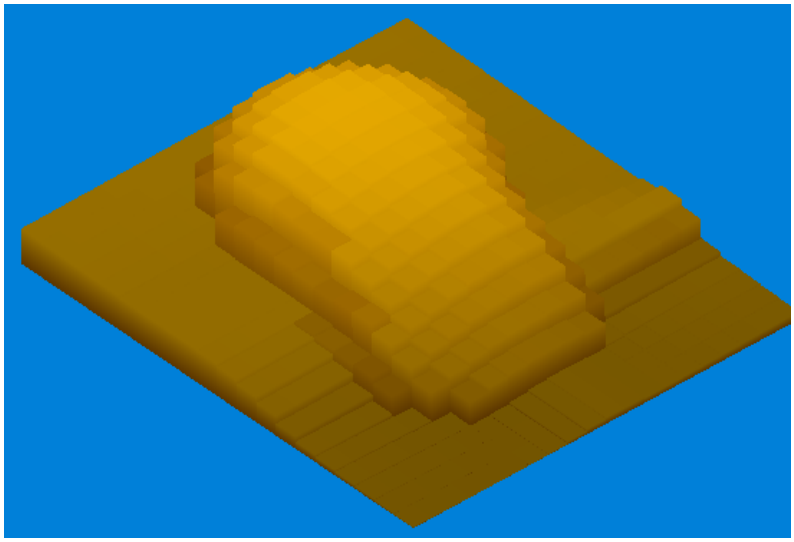


(a) Voxel model (b) Dexel model

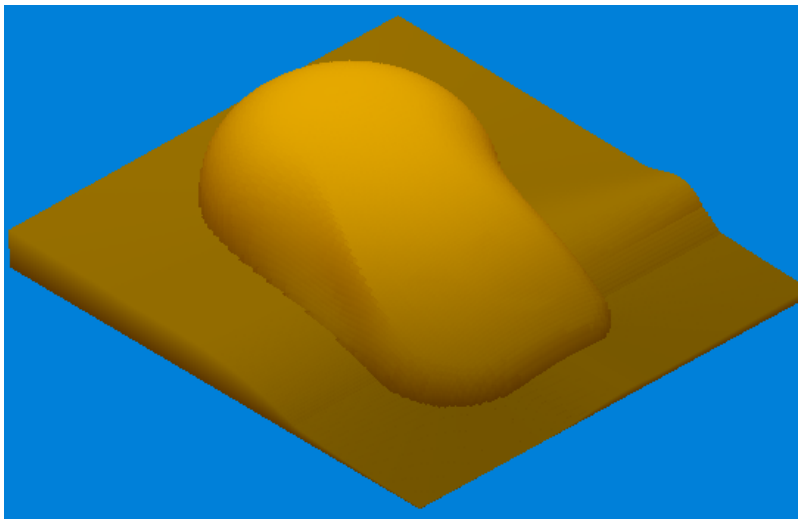
Figure 5.2. The comparison between Voxel model and Dexel model (same as **Figure 2.11**)

Constructing STL model from the resultant sculpted Dexel model turns out to be an interesting and challenging problem. In [Huang 1994], it was mentioned that Dexel model can be converted to two sets of equally spaced planar contours. It refers to some earlier work that the conversion from the planar contours to a triangular mesh, which amounts to an STL model, is possible [Meyer 1992]. However, by using the planar contours as the intermediate representation, it loses the topology information inherent in the original Dexel model. Hence, when there are several contours in a plane, the algorithms of conversion from planar contours to triangular mesh fail easily, because most of the time it is hard to judge which two contours on adjacent layers should be connected. In other words, it assumes perfect coherency between adjacent

layers, which is not necessarily true during virtual sculpting or NC simulation and verification processes. In [Konig Web], the condition when there is only one Dexel in each grid point was discussed. This condition rarely happens in virtual sculpting or 5-axis NC simulation. In [O'Connell 1993], solid models are constructed from 3-axis NC simulation by processing NC codes step by step. In [Roy 1999], geometric models are constructed from NC codes in an extended-Octree-based system. In [Leu 2002], they also mentioned this conversion problem.



(a) Dexel representation of a mouse model with a sparse density of 20x20



(b) Dexel representation of a mouse model with a density of 150x150

Figure 5.3. Dexel representation of a mouse model with different densities

In this chapter, the problem of converting Dexel volume models to STL surface models is analyzed and a marching algorithm is proposed as a solution. The remaining of this chapter is organized as follows. Section 5.2 briefly narrates the data structure of STL surface model and Dexel volume model. Section 5.3 describes how the Dexel volume model comes into being initially in the virtual sculpting system. Section 5.4 analyzes the problem of constructing STL model from Dexel model in details and proposes a marching algorithm to solve the problem. Implementation and experiment results are presented in Section 5.5. Conclusion is made and some important future work is discussed in Section 5.6.

5.2 STL and Dexel Representation of Solid Geometry

The data representations of STL polyhedral surface model and Dexel volume model are introduced first, to facilitate the discussion of conversion between them. STL model is basically a soup of triangular facets (Figure 5.1). Topology information, which defines the linkage between neighborhood triangles, has to be reconstructed from the original STL file. Since the number of triangles in a STL file is unknown at the beginning of reading, a dynamic memory allocation needs to be taken into account. An STL model may be recorded as a collection of triangular facets (triangles) and vertices as shown in Figure 5.4.

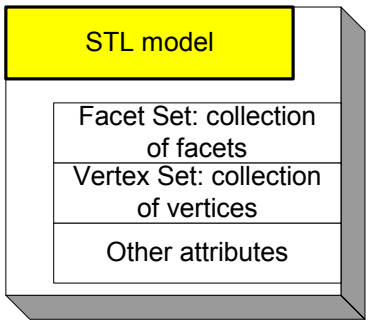


Figure 5.4. The data structure of STL surface model

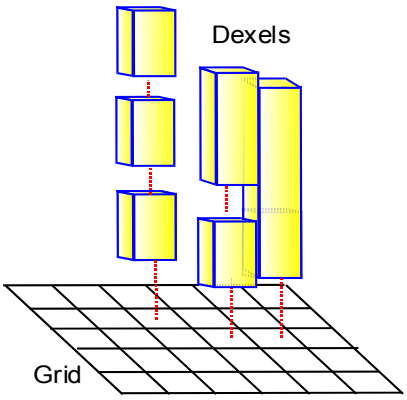


Figure 5.5. The linked list data structure of Dexel model

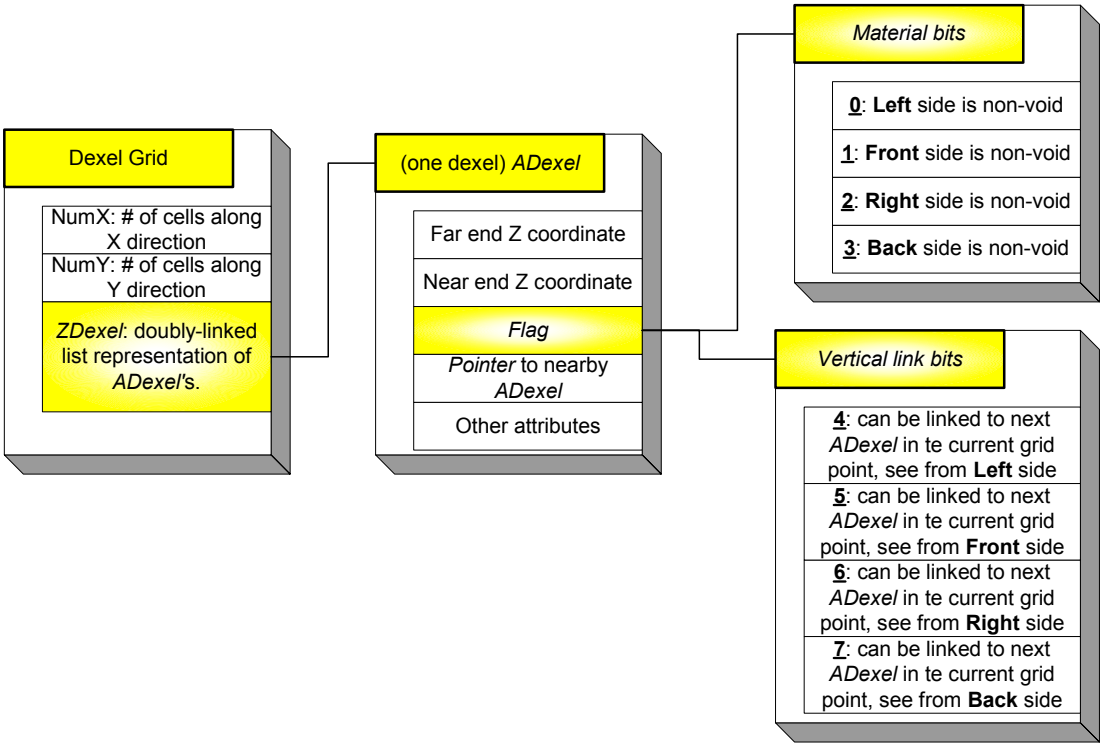


Figure 5.6. The data structure of Dexel volume model

Dexel model has been introduced in Chapter 2 and more details can be found in Chapters 3 and 4. In data structure, a Dexel volume model is recorded as a regular 2D grid, with each grid point associated with a linked list of Dexel elements (Figure 5.5). For simplicity, only a few sample Dexel linked list are illustrated in Figure 5.5. A Dexel model is a Dexel grid with *NumX* by *NumY* grid points, as shown in Figure 5.6. Without lose of generality, it can be assumed that the 2-D grid is on the X-Y plane and Dexels are extended along the Z axis direction. On each grid point, there is

a doubly-linked list with Dexel elements, which is defined as *ADexel* (See Figure 5.6). The detailed definition of *ADexel* is shown in the middle of Figure 5.6. It records the Z coordinates of the far end and the near end of a Dexel. There are also some pointers which are used to record the topology link information between the adjacent Dexel elements. Besides, it has a member named *Flag* (Figure 5.6). This *Flag* is a 32-bit long integer. It is used to track the information during the conversion from STL model to Dexel model. *Material bits* and *Vertical link bits* are two kinds of most important information recorded during the constructing process (Figure 5.6). It will be explained with details in Section 5.4.

5.3 Dexel Volume Models Generated by Virtual Sculpting

For haptic virtual sculpting, Dexel volume models are used as the in-process models for the interactive modification. During the virtual sculpting process, the swept volume of the virtual tool is subtracted from the virtual volume block represented in Dexel volume model format, if the tool carves into the block [Zhu 2003c]. Similar resultant Dexel volume model can be generated from multi-axis NC simulation processes, where the tool movement is pre-defined in NC codes [Huang 1994], while in virtual sculpting the tool movement is determined in real-time. Several challenges of the resultant Dexel volume model are observed as follows (see Figure 5.7, for more practical examples, see Section 5.5):

- The shape of the resultant Dexel volume model is globally arbitrary with local coherency. Not all parts of the Dexel model are necessarily connected, as shown in Figure 5.7.
- For each disjointed part of the Dexel model, it forms a closed object. In this sense, the Dexel model can be imagined as several ‘islands’ floating around the space, while these ‘islands’ are arbitrary in shape (Figure 5.7).
- For each of such ‘islands’, it can be approached from six directions, namely left and right, top and bottom, and front and back. Since the 2-D

grid of the Dexel model is assumed to be on X-Y plane, the top and bottom directions have the priority if STL model is meant to be constructed from the resultant Dexel volume model.

- Besides the top and bottom surface, the other surface should be constructed at the transition area where a solid region enters a void region, or *vice versa*.

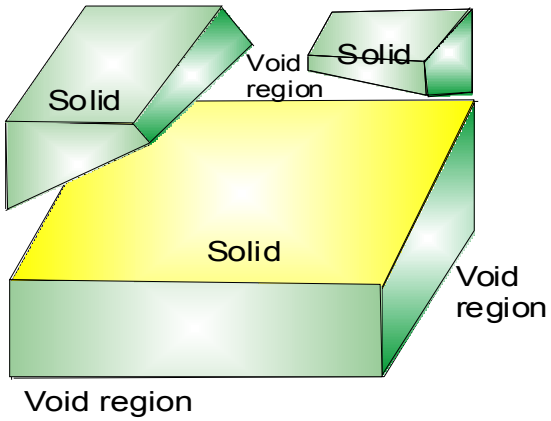


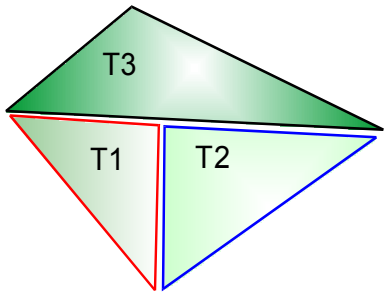
Figure 5.7. An example Dexel model with disjointed components

The conversion from Dexel volume model to STL model is a challenging problem because of the concerns mentioned above. Although seminal work and abundant literature can be found on generating triangular meshes from Voxel model [Lorensen 1987], we are not aware of any research papers that seriously provide the detailed solution to the problem of conversion from Dexel model to STL model. *Marching Cube* algorithm presented in [Lorensen 1987] is applicable only to Voxel models and also may result in some topological errors [Gelder 1994].

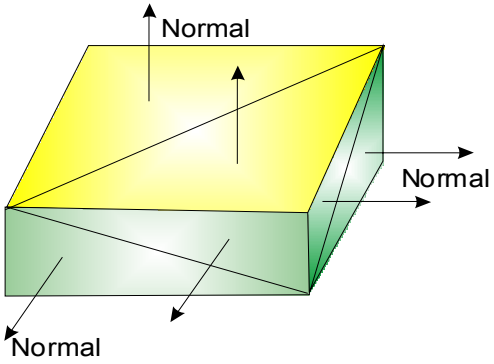
In this chapter, our investigation are focused on the converting the disjoined complex Dexel volume models into STL polyhedral surface models for CAD/CAM or RP systems. For the resultant STL model, the objectives of the conversion are shown as follows:

- A closed region should be *water-tight* without gaps or holes;
- Each triangle has correct orientation with its normal vector pointing outward of the model (instead of pointing inward) (Figure 5.8(a));

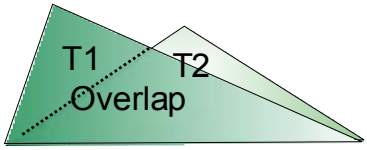
- There are no T junctions (Figure 5.8(b));
- No overlap / duplicate triangles (Figure 5.8(c)).



(a) T Junction not allowed



(b) Normal direction pointing outward



(c) Overlap triangles not allowed

Figure 5.8. Objectives in constructing STL surface models

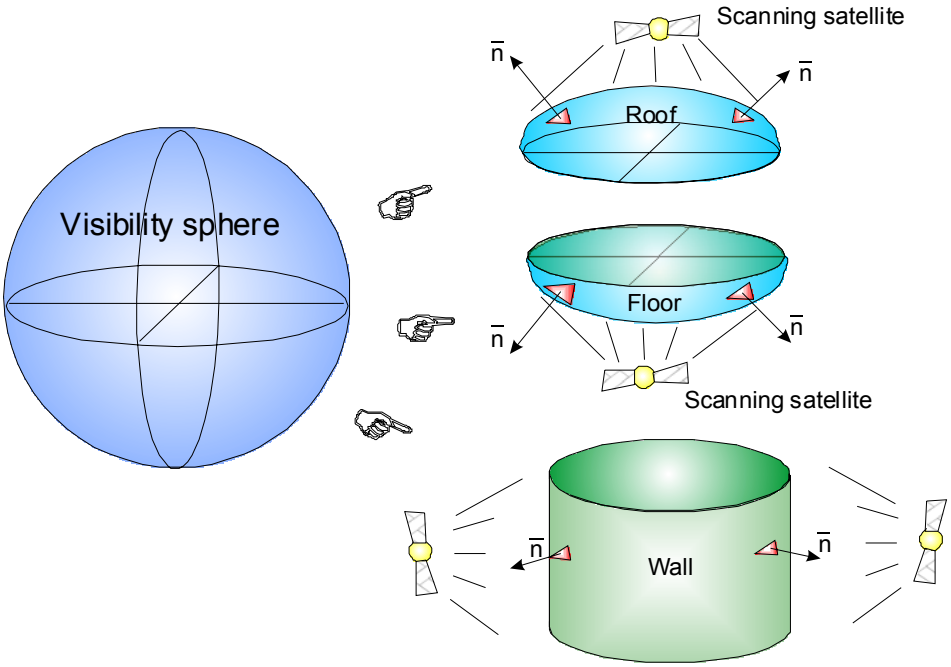
In the next section, this conversion problem is analyzed and a marching algorithm is proposed to solve the problem.

5.4 Marching Algorithm for Constructing STL Models from Dexel Volume Models

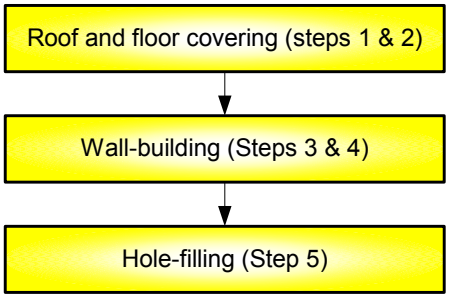
The concept of visibility sphere is first introduced to facilitate the explanation of the proposed marching algorithm (see Figure 5.9(a)). Imagine that the 3D Dexel

volume model is sitting in the middle of the visibility sphere, and a satellite is scanning the visibility sphere. If it scans from the top, the roof of model can be covered and thus constructed. If it scans from the bottom, the floor of model can be covered and thus constructed. If it scans from the sides, the walls of model can be covered and thus constructed. There may still be some gaps and holes up to this point. Thus gaps/holes needs to be filled. With the above objectives and concept in mind, the constructing approach has three phases conceptually, which are also analogous to building a house (Figure 5.9(b)):

- *Roof and floor covering*: construct facets visible from top or bottom
- *Wall-building*: construct facets visible from sides
- *Hole-filling*: construct facets to fill the holes or gaps



(a) Visibility sphere concept during the constructing process



(b) Three phases

Figure 5.9. Visibility sphere and the overview of the marching algorithm

It is noted that *Flag* in the *ADexel* structure (Dexel element) is used to track the information related to the current *ADexel* (see Figure 5.6). As an integer, *Flag* can take 32 bits. The right two shadowed boxes in Figure 5.6 show the most important *Flag* bits defined for each *ADexel* (Dexel element). These *Flag* bits are set in ***Steps 1 and 2*** (*roof and floor covering*). If *Material bits* of an *ADexel* are set, it means the Dexel is on the boundary of a surface and thus should be linked in ***Steps 3 and 4*** (*wall-building*). If *Vertical link bits* of an *ADexel* are set, it means it is somewhere on the boundary of a surface and thus should be linked in ***Step 5*** (*hole-filling*). The general algorithm is presented as follows (Figures 5.9 to 12 and also refer to Figure 5.6):

Step 1: Marching from front left corner of the 2D grid to the back right corner: connect *far end* pairs and *near end* pairs of adjacent *ADexel*'s, at the same time setting appropriate *Flag* bits (defined in Figure 5.6) for further processing. This would form the top and bottom surfaces (*roof and floor covering*) (see Figure 5.10).

Step 2: Marching from back right corner of the 2D grid to the front left corner: this time also sets some appropriate *Flag* bits. The *Flag* set in both ***Step 1*** and ***Step 2*** is to be used as criteria for constructing left, right, front and back surfaces in later steps (see Figure 5.10).

Step 3: Marching from front left corner of the 2D grid to the back right corner again: this time check the *Flag* bits, and form the left and front surfaces accordingly (*wall-building*) (see Figure 5.11).

Step 4: Marching from back right corner of the 2D grid to the front left corner: check the *Flag* bits and form the right and back surfaces accordingly (*wall-building*) (see Figure 5.11).

Step 5: Marching from front left corner of the 2D grid to the back right corner again: check if there is transition between a solid region and a void region and fill the holes/gaps accordingly (*hole-filling*) (see Figure 5.12).

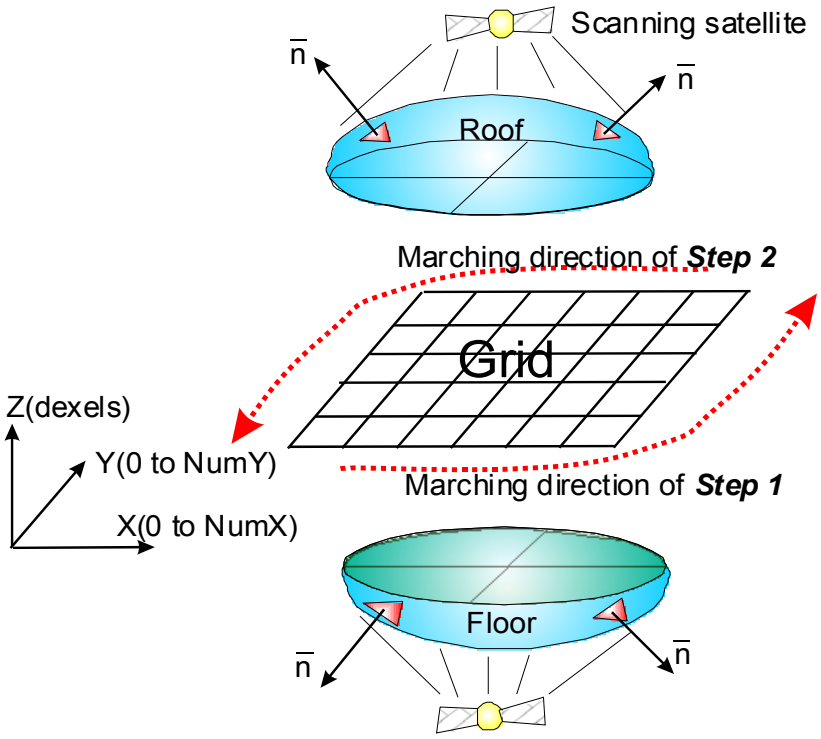


Figure 5.10. The marching directions during the 'roof and floor covering'

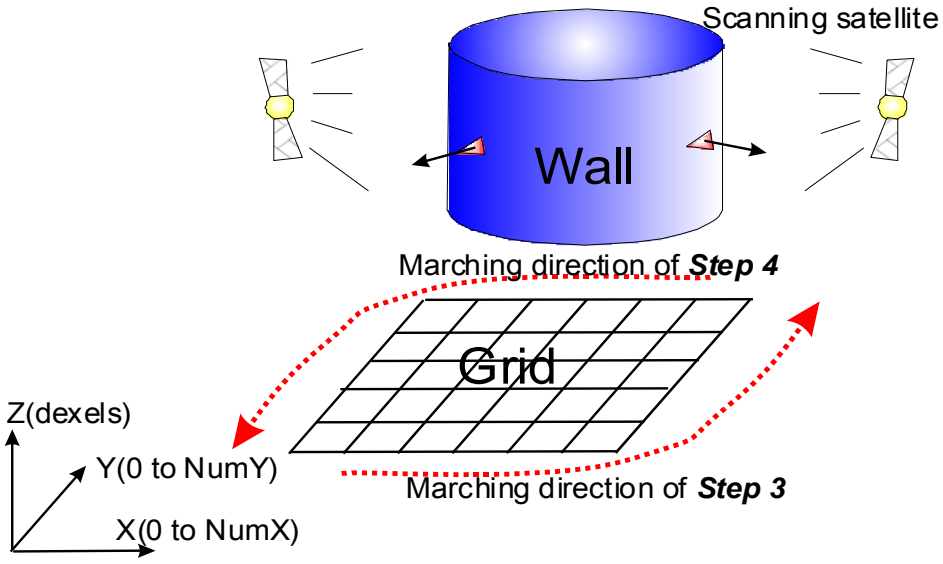


Figure 5.11. The marching directions during the 'wall-building'

For clarity, Figures 5.13 and 5.14 give an overview of the working principles of the algorithm by an example. Figure 5.13 shows an example Dexel model. The density of the Dexel model is based on a 150x150 grid. By using a haptic virtual sculpting system developed in our earlier work in [Zhu 2003a], the Dexel volume

model is converted to the STL model. Figure 5.14 shows how the resultant STL model look like after each step as described above. The result after executing ***Step 1*** is shown in Figure 5.14(a). It can be seen that the top and bottom surfaces came into being. The result after executing ***Step 3*** is shown in Figure 5.14(b). It can be seen that the left and front surfaces have been formed. However, the right and back surface are still non-existing (Figure 5.14(c)) and holes are unfilled (Figures 5.14(b) and (d)). The result after executing ***Step 4*** is shown in Figure 5.14(e). It can be seen that the right and back surfaces have been created. After ***Step 5***, the final resultant STL model is shown in Figure 5.14(f). The final STL model is a water-tight model without holes. The detail concerning each step is presented in the following sub-sections.

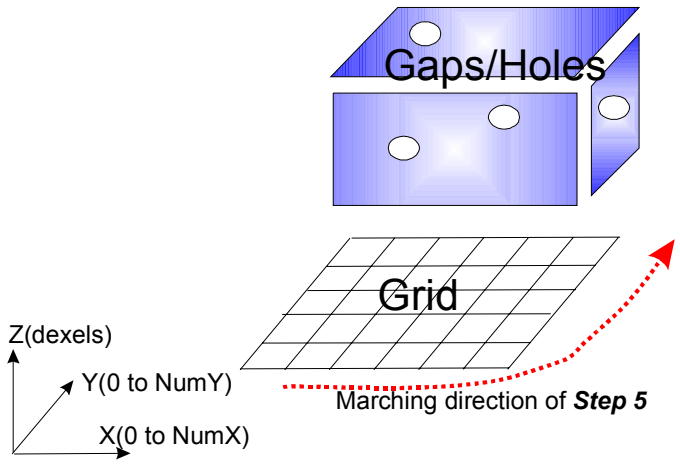


Figure 5.12. The marching directions during the 'hole-filling'

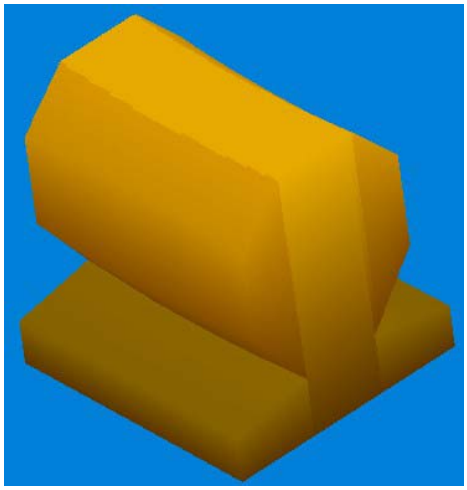
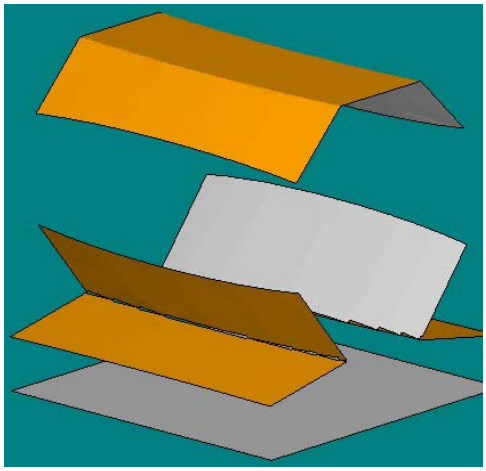
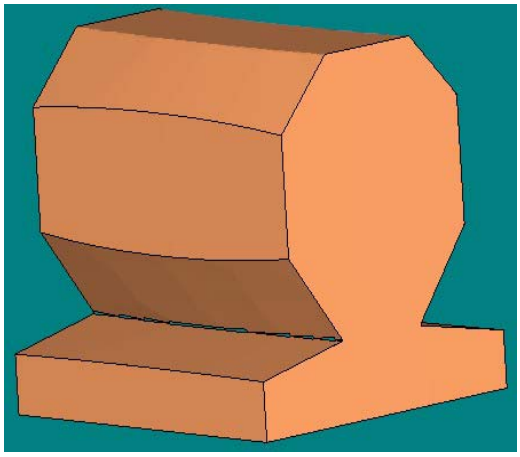


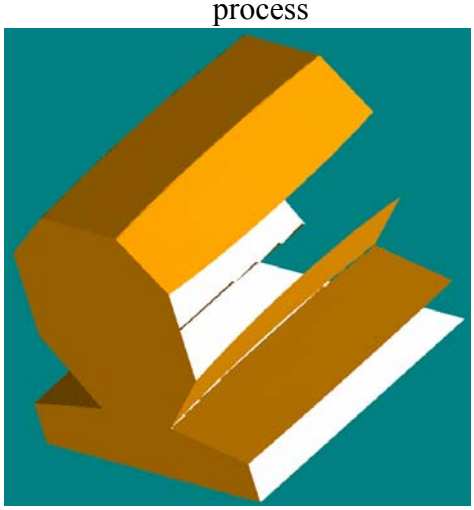
Figure 5.13. An example Dexel model converted from an STL model



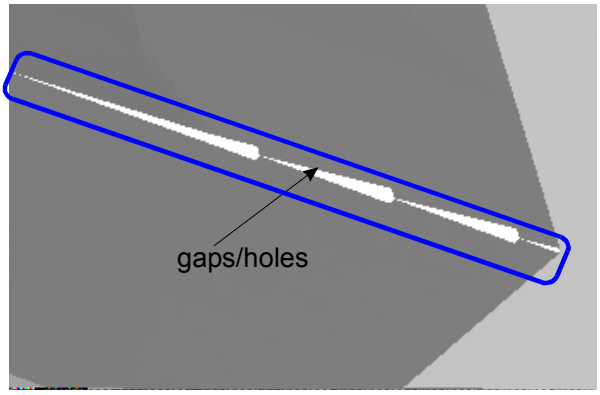
(a) After Step 1 of conversion process



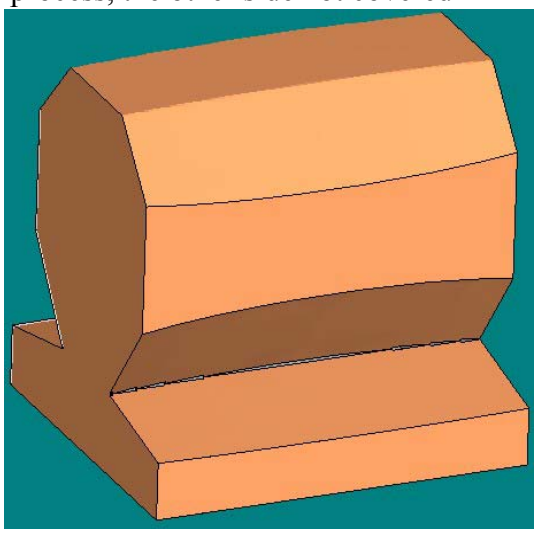
(b) After Step 3 of conversion process



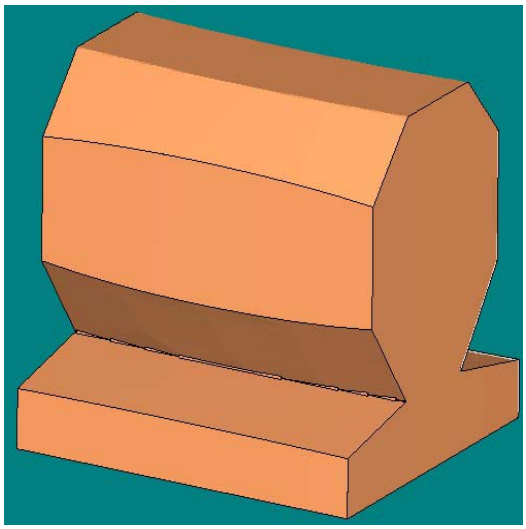
(c) After Step 3 of conversion process, the other side not covered



(d) After Step 3 of conversion process, gaps/holes not filled



(e) After Step 4 of conversion process, the other side is covered now



(f) After Step 5 of conversion process, everything is fine

Figure 5.14. An illustrative example of conversion from Dexel model to STL model

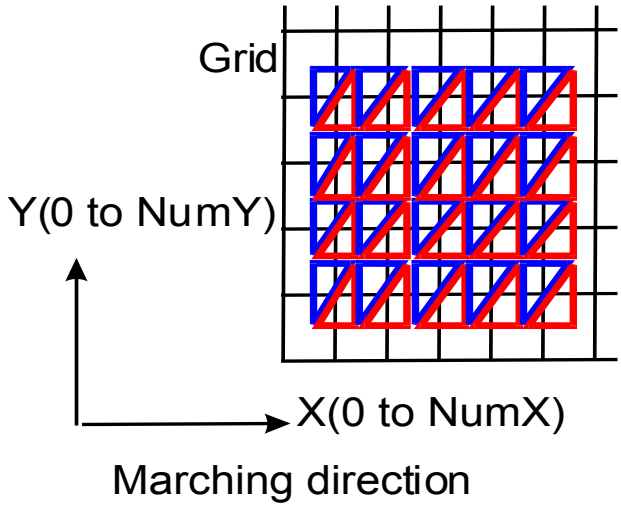


Figure 5.15. Marching from corner (0,0) to corner (*NumX*, *NumY*), and form triangles for the top and bottom surface if applicable

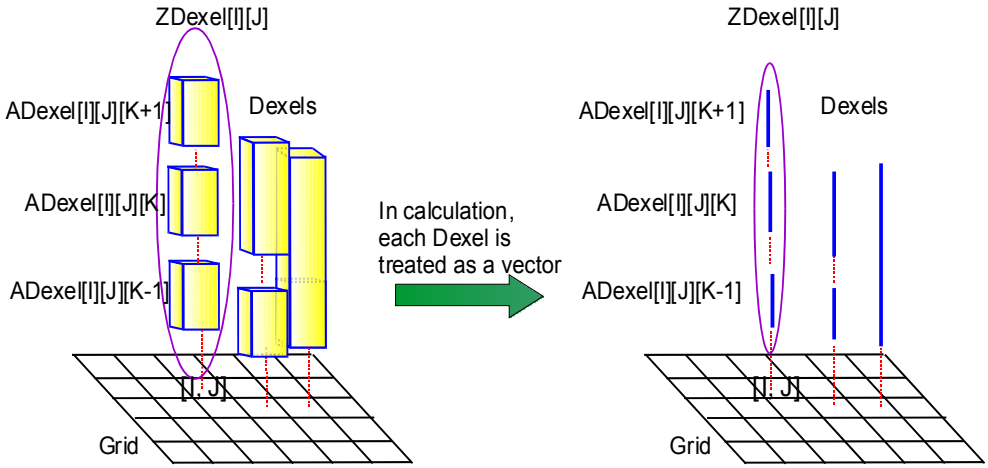


Figure 5.16. *ADexel[I][J][K]* is a Dexel element and and *ZDexel[I][J]* is an array of Dexel elements on a Dexel grid point; In calculation, each Dexel is treated as a vector

5.4.1 Roof and floor covering (Steps 1 & 2)

In ***Step 1***, the top and bottom surfaces are constructed. This step compares the adjacent Dexels and try to form triangles by connecting the center of adjacent Dexels. As shown in Figure 5.15, the marching direction is from front left corner (0, 0) to back right corner (*NumX*, *NumY*) and triangles are formed if applicable. By definition, *ADexel[I][J][K]* is a Dexel element and *ZDexel[I][J]* is an array of Dexel elements on a Dexel grid point (see Figure 5.16). In calculation, each Dexel element is treated

as a vector. The flowchart of *roof and floor covering algorithm* is shown as follows (also see Figure 5.20):

1) *Check out one Dexel element*: get one Dexel element *ADexel[I][J][K]* from the current Dexel list *ZDexel[I][J]*, where $I \in [0, NumX]$, $J \in [0, NumY]$ and K is the sequence number of this *ADexel* (Dexel element) in the current Dexel list.
2) *Check out overlap Dexel elements in adjacent Dexel lists*: check three adjacent Dexel lists *ZDexel[I+1][J]*, *ZDexel[I+1][J+1]*, and *ZDexel[I][J+1]* with *ADexel[I][J][K]*. Any *ADexel*'s in the three adjacent Dexel lists overlapping with *ADexel[I][J][K]* are checked out and recorded in three temporary lists respectively (Figure 5.17). Meanwhile, the *Material bits* (see Figure 5.6) for these *ADexel*'s in the three temporary lists are set accordingly. Denote the three temporary lists as *TZDexel[I+1][J]*, *TZDexel[I+1][J+1]*, and *TZDexel[I][J+1]*.
3) *Check for validity of linking between overlap adjacent Dexel elements*: if there is only one Dexel in the current Dexel list *ZDexel[I][J]*, then the far ends of *ADexel[I][J][K]* can be legally linked with the far ends of the farthest *ADexel* in adjacent *TZDexel* lists, to form triangles as shown in Figure 5.18. The same linkage applies to the near ends. **However**, the check in the previous step only perform the overlap test with *ADexel[I][J][K]*. The previous *ADexel[I][J][K-1]* and next *ADexel[I][J][K+1]* should also be considered. For example, in Figure 5.19, *ADexel[I][J][K]* check out three *ADexel*'s in one of its adjacent lists but its far end is blocked by *ADexel[I][J][K+1]*. Hence the previous and next *ADexel* (Dexel element) are used to check for validity of linking. *Material bits* and *Vertical link bits* (see Figure 5.6) are set as appropriately. If *TZDexel* lists are checked to be legal links with *ADexel[I][J][K]*, then triangles are formed as part of the top or bottom surface (see Figure 5.14(a)).

4) *Record topology link information between linked Dexel elements*: during the process of forming triangles, topology link information between the adjacent *ADexel*'s is recorded with Pointers as described earlier in Figure 5.6. This information is used later in ***Steps 3, 4*** and ***5***.

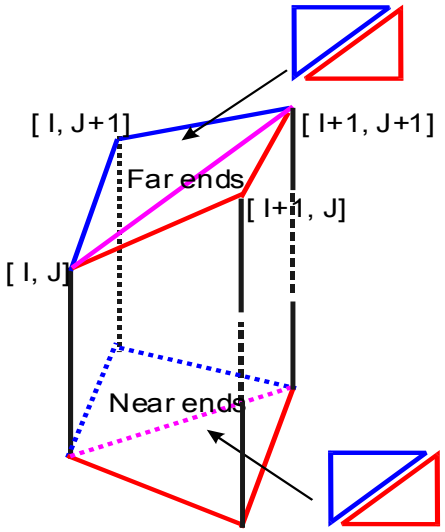


Figure 5.17. Check adjacent Dexel lists for potential link of triangles (for simplicity, the Dexels are represented with lines)

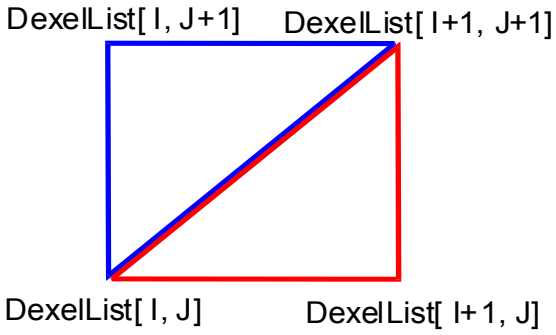


Figure 5.18. Link adjacent Dexel lists' far end or near end, respectively to form triangles for the top and bottom surfaces

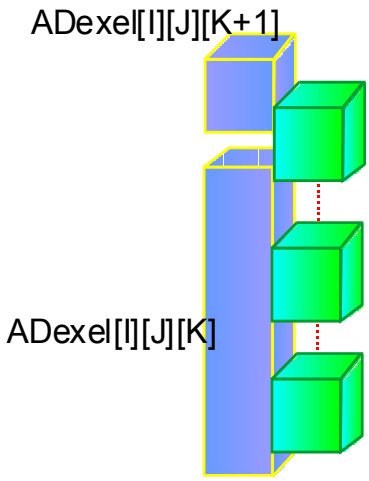


Figure 5.19. Current Dexel element *ADexeL[I][J][K]* should be further checked with previous Dexel element *ADexel[I][J][K-1]* and next Dexel element *ADexel[I][J][K+1]* for valid linking

In ***Step 2***, it doesn't form any new triangle but just updates the topology information of the converted triangular facets; otherwise it is similar to ***Step 1***. Basically it marches from back right corner (*NumX*, *NumY*) to front left corner (0, 0) and sets the appropriate *Material bits* and *Vertical link bits* for *ADexel*'s (Figure 5.6). At the same time, topology link information between the adjacent *ADexel*'s is recorded with Pointers as described in Figure 5.6.

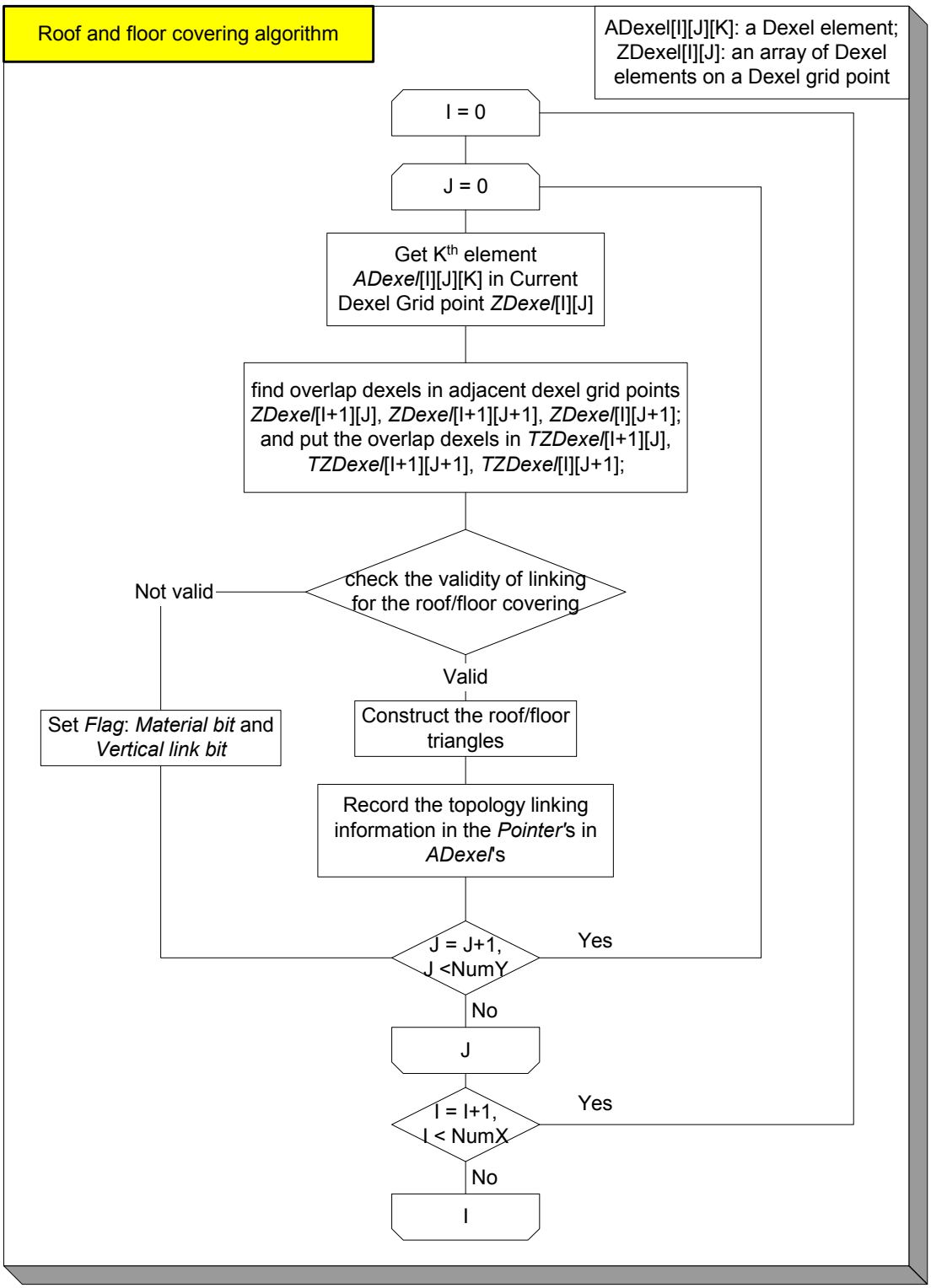


Figure 5.20. Flowchart of 'roof and floor covering algorithm'

5.4.2 Wall-building (Steps 3 & 4)

In ***Step 3***, it marches from front left corner (0, 0) to back right corner (*NumX*, *NumY*) and try to construct the front and left surface (see Figure 5.14(b)). The links

between the far end and near end of *ADexel*'s form the triangles for the front and left surface. This is analogous to building vertical walls for a house. The flowchart of *wall-building algorithm* is shown as follows (also see Figure 5.24):

1) *Check out one Dexel element*: get one *ADexel[I][J][K]* from the current Dexel list *ZDexel[I][J]*, where $I \in [0, NumX]$, $J \in [0, NumY]$ and K is the sequence number of this *ADexel* (Dexel element) in the current Dexel list.
2) *Check out overlap Dexel elements in adjacent Dexel lists*: check two adjacent Dexel lists *ZDexel[I+1][J]* and *ZDexel[I][J+1]* with *ADexel[I][J][K]*. Any *ADexel*'s in the two adjacent Dexel lists overlapping with *ADexel[I][J][K]* are checked out and recorded in two temporary lists respectively (Figure 5.21). Denote the two temporary lists as *TZDexel[I+1][J]* and *TZDexel[I][J+1]*.
3) *Linking overlap adjacent Dexel elements in **Front** wall*: construct the **Front** wall if there exists *TZDexel[I+1][J]* and *ADexel[I][J][K]*'s *Material bits* indicates this Dexel element is a wall element (transition from void region to solid region). Figure 5.21 illustrates the process of 'building walls' on the front surface (**Front** wall): current Dexel element *ADexel[I][J][K]* may be overlapped with several *ADexel*'s on the right and hence several triangles may be performed. It shows there are two *ADexel*'s to the right of the current Dexel element *ADexel[I][J][K]*. This kind of connection method is to avoid any T junction with other triangles (see Figure 5.8(c)).
4) *Linking overlap adjacent Dexel elements in **Left** wall*: Construct the **Left** wall if there exists *TZDexel[I][J+1]* and *ADexel[I][J][K]*'s *Material bits* indicates this Dexel element is a wall element (transition from void region to solid region).

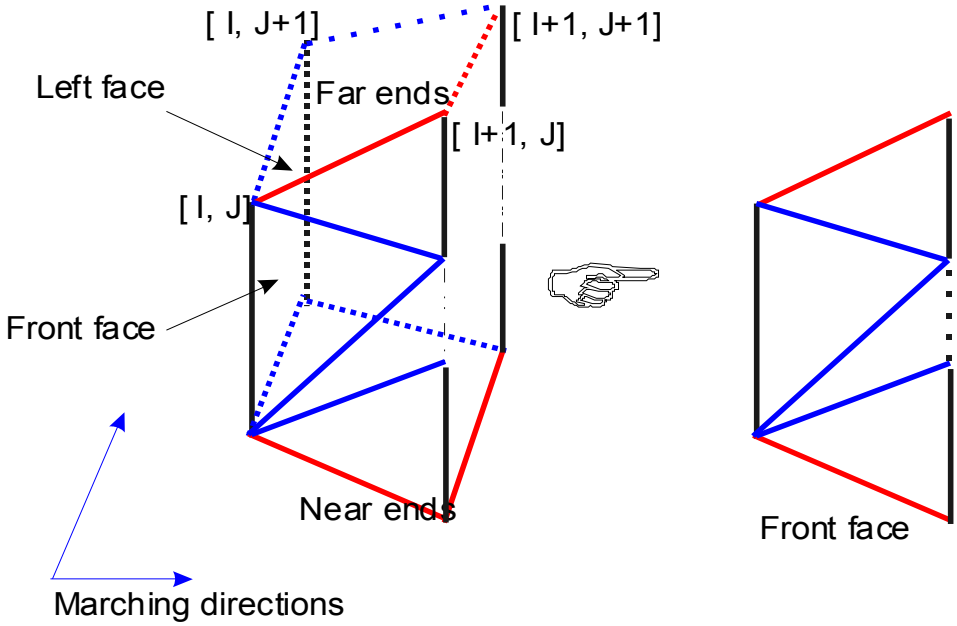
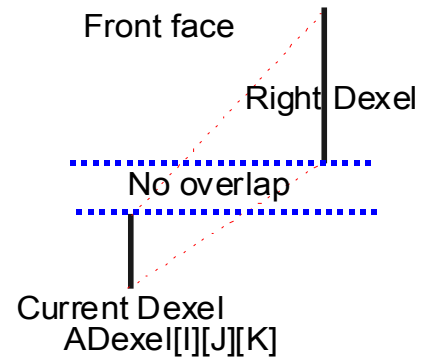


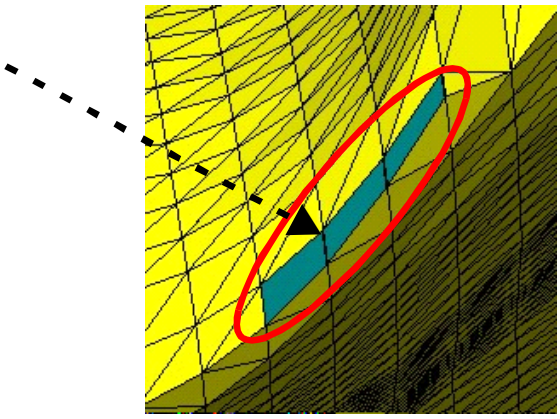
Figure 5.21. 'Wall Building': front surface triangles are being created

A special condition needs to be considered: there may be no overlapping between adjacent *ADexel*'s, while they are topologically linked (Figures 5.22(a) and (b)). This could happen in highly curved region, when the Dexel grid density is not high enough to ensure local coherency of the topology. Figure 5.22(b) shows an example for this case. Hence overlap test doesn't cover this case. Fortunately, the topology information has been collected in the previous steps. If there is no overlapping *ADexel*'s found and based on the collected topology information, *ADexel[I][J][K]* can be linked to the topological neighbor *ADexel*'s, as shown in Figure 5.23. The regional defects as in Figure 5.22(b) is fixed as in Figure 5.22(c).

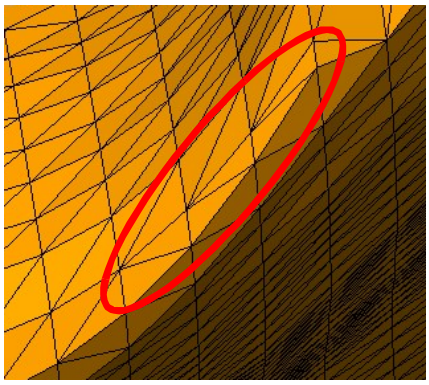
Step 4 is similar to ***Step 3***, except that it is marching from back right corner (*NumX*, *NumY*) to front left corner (0, 0) and try to construct the back and right surface (see Figure 5.14(c, e)).



(a) No overlapping but could be topologically connected



(b) No overlapping but should be topologically connected, during 'Wall building'



(c) Topologically connected triangles

Figure 5.22. An example for special condition during 'Wall Building': no overlapping but topologically connected

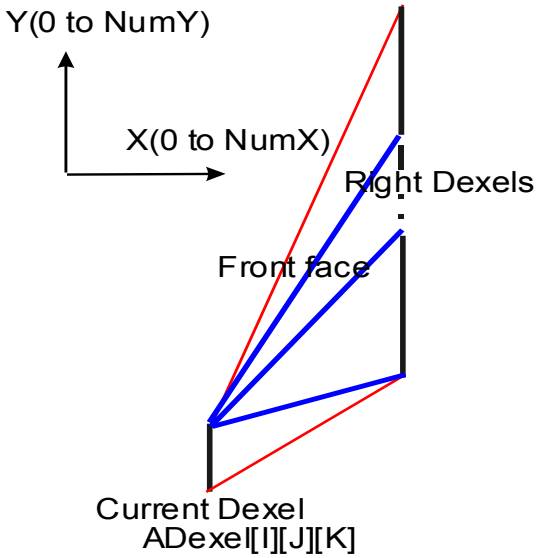


Figure 5.23. Illustration for special condition during 'Wall Building': no overlapping but topologically connected

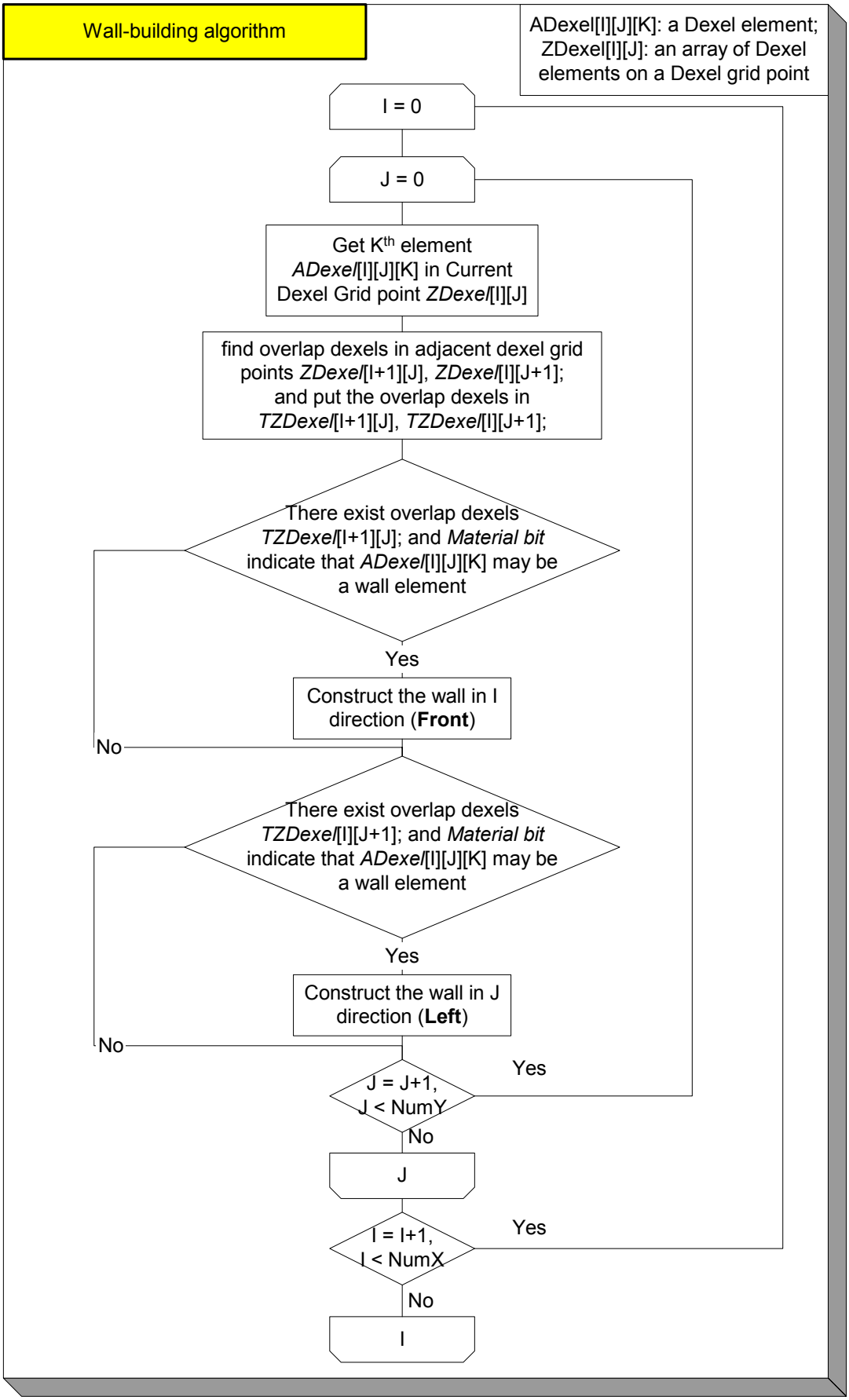


Figure 5.24. The flowchart of 'wall-building algorithm'

5.4.3 Hole-filling (Step 5)

Step 5 tries to fill the holes/gaps left after the previous steps. These holes/gaps are formed by two sequential *ADexel*'s (Dexel elements) in the same Dexel list and both such *ADexel*'s are on the boundary of surfaces, through which a solid region transits into a void region or *vice versa* (see Figure 5.14(d)). '*Gap*' is the name given to the empty space (including holes and gaps) between two sequential *ADexel*'s in the same Dexel list. The task is to link these gaps to fill the holes. *Vertical link bits* in an *ADexel*'s *Flag* are used to determine whether there is *Gap* on the far end of the current *ADexel* (Figure 5.6). The flowchart of *hole-filling* algorithm is shown in Figure 5.26. Its explanation is as follows:

1) *Check out two successive Dexel elements*: get one *ADexel[I][J][K]* and next one *ADexel[I][J][K+1]* from the current Dexel list *ZDexel[I][J]*, where $I \in [0, NumX]$, $J \in [0, NumY]$ and K is the sequence number of *ADexel* in the current Dexel list:
2) *Check out overlap Gaps in adjacent Dexel lists*: Check two adjacent Dexel lists *ZDexel[I+1][J]* and *ZDexel[I][J+1]* with *ADexel[I][J][K]*. Any valid *Gap* in the two adjacent Dexel lists overlapping with the *Gap* between *ADexel[I][J][K]* and *ADexel[I][J][K+1]* are checked out and recorded in two temporary lists respectively. Denote the two temporary lists as *GZDexel[I+1][J]* and *GZDexel[I][J+1]*.
3) *Fill Gaps*: Link adjacent valid *Gap*'s to form triangles to fill the holes. The condition when one *Gap* corresponds to multiple adjacent *Gap*'s should be considered, as shown in Figure 5.25.
4) *Fill diagonal Gaps, if exist*: If, as in Figure 5.18, only one of the pair triangles is constructed, then there are diagonal *Gap*'s, which should also be connected. Under such condition, the similar check and link method as in 2) and 3) may be applied to *ZDexel[I+1][J+1]*:

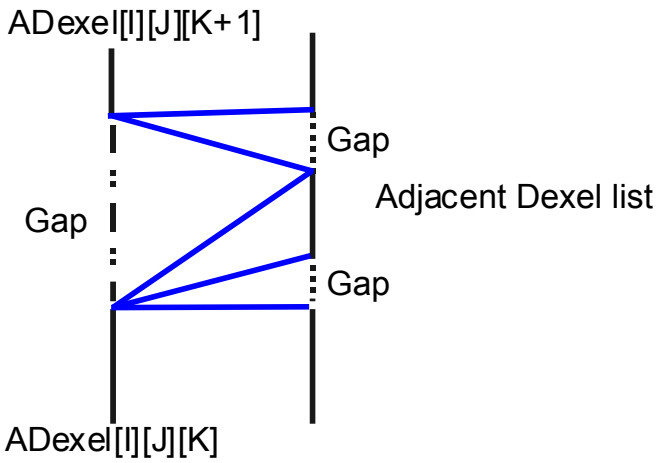


Figure 5.25. Connect *Gap*'s to fill holes (*ADexeL[I][J][K]*: Current Dexel element, *ADexeL[I][J][K+1]*: Next Dexel element)

5.4.4 Optimizing the converted STL models

The above marching method generated large number of triangles. The number of triangles can be reduced in many cases. Hence, it is desirable to apply some triangular mesh decimation algorithms to reduce the number of triangles and smooth the final surface [Schroeder 1992]. Efficient triangle decimation methods with different objectives have been reported in several past chapters as in [Schroeder 1992] and [Garland 1997], *etc*.

The theoretical description is thus far described. Next section the results of the software implementation and experiments are to be presented.

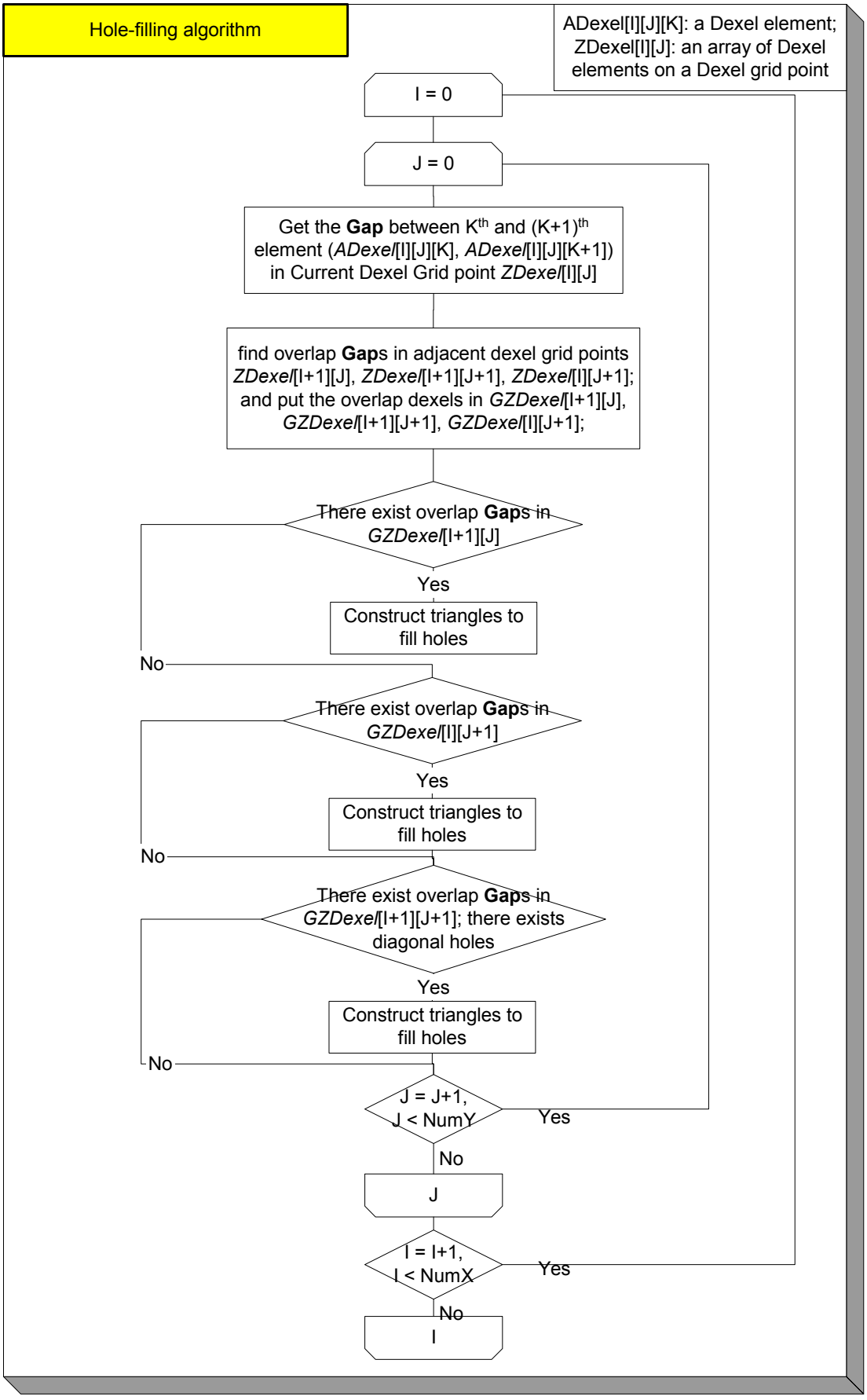


Figure 5.26. The flowchart of 'hole-filling algorithm'

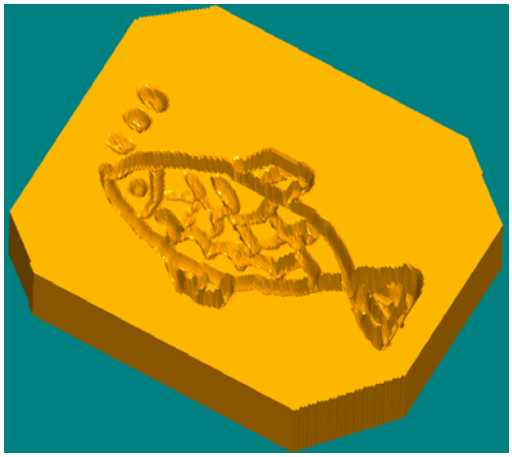
5.5 Implementation and Examples

The presented method and algorithms have been implemented on a dual 2.4GHz CPU workstation, with Visual C++ and OpenGL®, as part of our developed haptic virtual sculpting system [Zhu 2003a and Zhu 2003b]. All the example models use a Dexel grid density of 150x150. With this density and computer configuration, most of the STL models take negligible time to be converted into Dexel models and it usually takes less than one second to convert a Dexel model to an STL model. The conversion time varies with the model complexity and the triangles generated. A medium complex Dexel model with a density of 150x150 generates about 80,000 to 200,000 triangles.

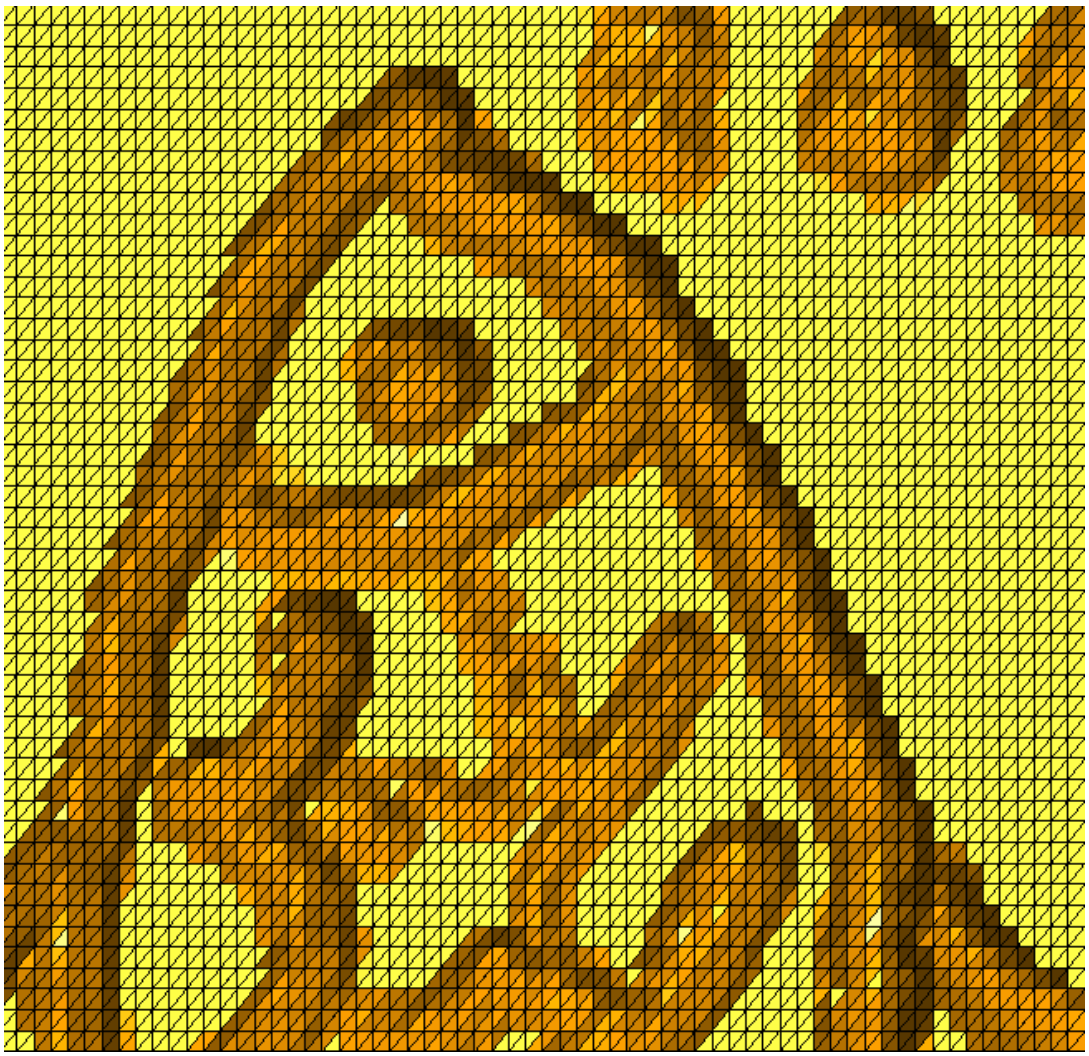
Figure 5.27(a) shows a fish Dexel model generated from virtual sculpting. The Dexel-based virtual sculpting system has been developed in our earlier work in [Zhu 2003a]. It uses a 5-DOF haptic (force-feedback) device as an input to create some intuitive design. The fish Dexel model is converted to an STL model as shown in Figure 5.27(b). Details of the fish STL model are shown in Figure 5.27(c).

Figure 5.28(a) shows a Dexel volume model generated from NC simulation program developed in our lab. Besides the surface generated, there are some floating rest material in the space. The whole Dexel model is converted to an STL model after simulation, as shown in Figure 5.28(b). Figure 5.28(c) shows the close-up view of the machined surface. More examples are presented in Chapter 6.

(a) A fish Dexel model from virtual sculpting

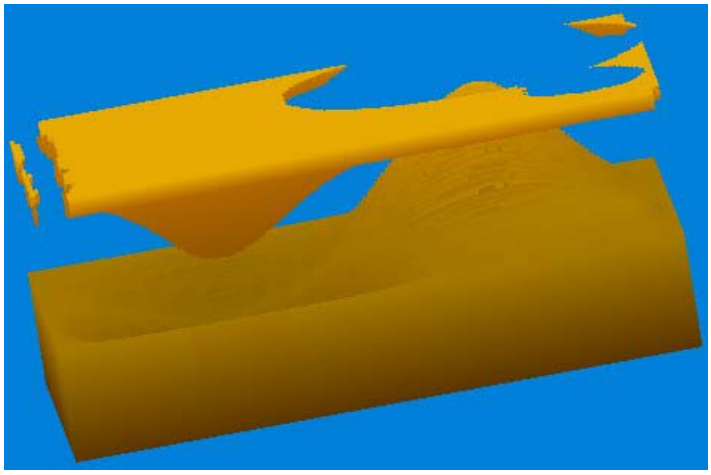


(b) The fish Dexel model is converted to an STL model

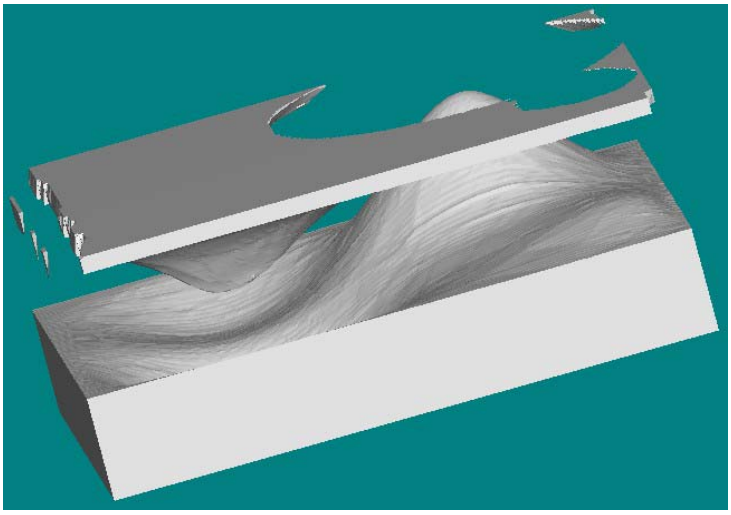


(c) The detail of the fish STL model

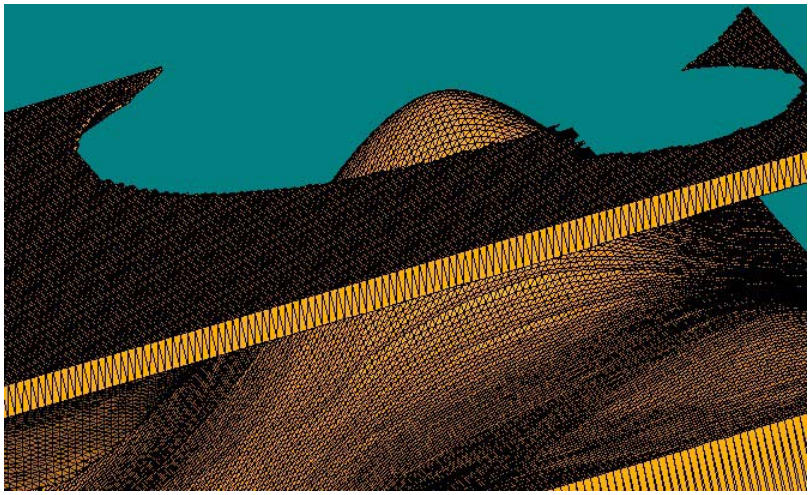
Figure 5.27. Conversion of an fish Dexel model of Virtual Sculpting to an STL model



(a) The NC simulation result for a freeform surface: a Dexel model



(b) The resultant simulation Dexel model has been converted to an STL model.



(c) The details of the converted STL simulation model

Figure 5.28. The conversion of an NC simulation resultant Dexel model to an STL model

5.6 Summary

In this chapter, a marching algorithm on constructing STL polyhedral surface model from Dexel volume model is presented. The shapes of Dexel volume models from virtual sculpting system are globally arbitrary with local coherency. STL surface model needs to be constructed from Dexel volume model for further processing in available CAM (Computer-aided manufacturing) or RP (Rapid prototyping) systems. The constructing process of STL model is analogous to building a house. It is composed of 3 sub-algorithms: roof and floor covering, wall-building and hole-filling algorithms. Algorithms have been developed and implemented with C++ and OpenGL based on the theoretical analysis. Examples from haptic virtual sculpting system and NC simulation processes are used to prove the effectiveness of the algorithms.

Some future research work needs to be done to improve the current work. The marching algorithm on converting Dexel mode to STL model is generally effective on most of the tested examples. However, we did notice that it leaves some cracks and holes when applied to some very complex models. More work in Topology may be needed to identify the reason [Gelder 1994]. On the other hand, these defects on the generated model are in a very small percentage. It may be fixed interactively with some commercial software. The algorithms generated a lot of triangles, which may not be necessary. Hence, it is helpful to apply some decimation algorithms to reduce the number of triangles and smooth the final surface [Schroeder 1992, Garland 1997]. It is especially desirable when an STL model is output from a virtual sculpting system.

Chapter 6

Computer Implementations and Examples

In the previous chapters, the methodology of the virtual sculpting and machining planning system has been presented. Some of the computer implementation, experiments and examples are also presented. This chapter discusses more implementation issues and practical examples generated from the developed virtual sculpting and machining planning system.

6.1 Implementations and Examples

A practical shoe sole model was used as an illustrative example for 5-axis *pencil-cut* tool path generated by the developed machining planning system. Figure 6.1 shows the process of determining tool orientation. Figure 6.2 shows a rendered view of the generated tool path with tool orientations.

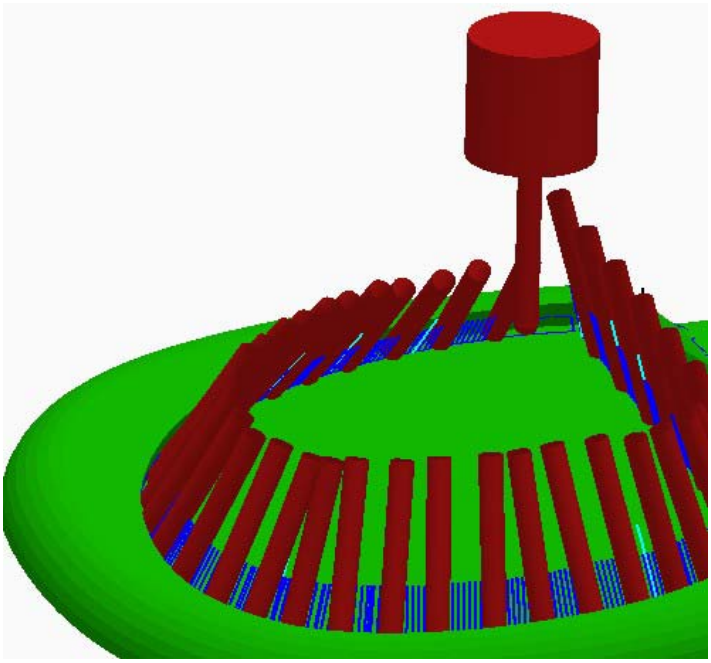


Figure 6.1. Tool orientation selection process for a shoe sole model

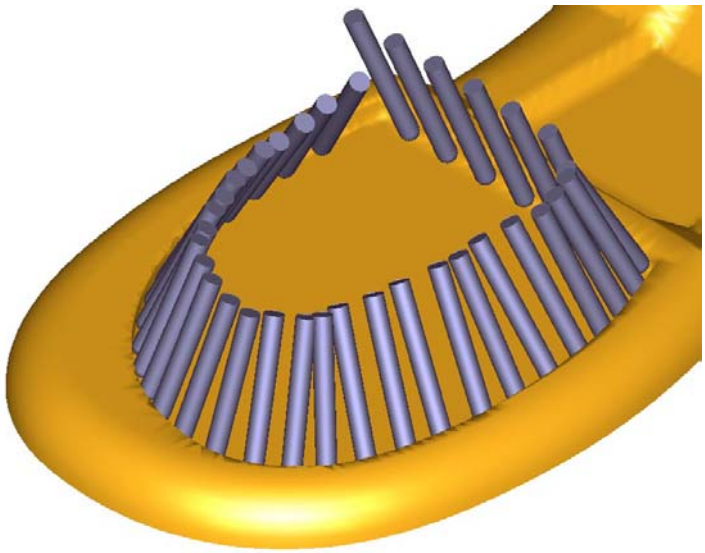


Figure 6.2. A rendered view of selected tool orientations for a shoe sole model

The NC codes generated from the haptic 5-axis *pencil-cut* system were simulated by commercial NC simulation software in our lab. Figure 6.3 shows the example part surface after finishing and before pencil-cut. Figure 6.4 shows the example part surface during 5-axis *pencil-cut* operation. The rest material near the shared sharp edges is being removed by the pencil-cut tool path, as shown in Figure 6.5. Figures 6.6, 6.7 and 6.8 show how the rest material on the side of the shoe model is removed. The NC machining simulation results shows the effectiveness of the proposed 5-axis pencil-cut.

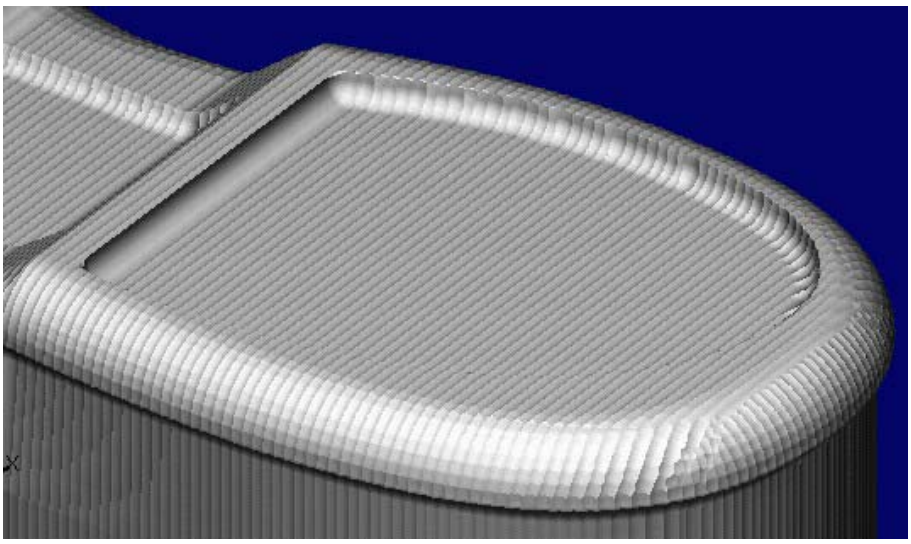


Figure 6.3. Shoe sole model after finishing without pencil-cut (top surface)

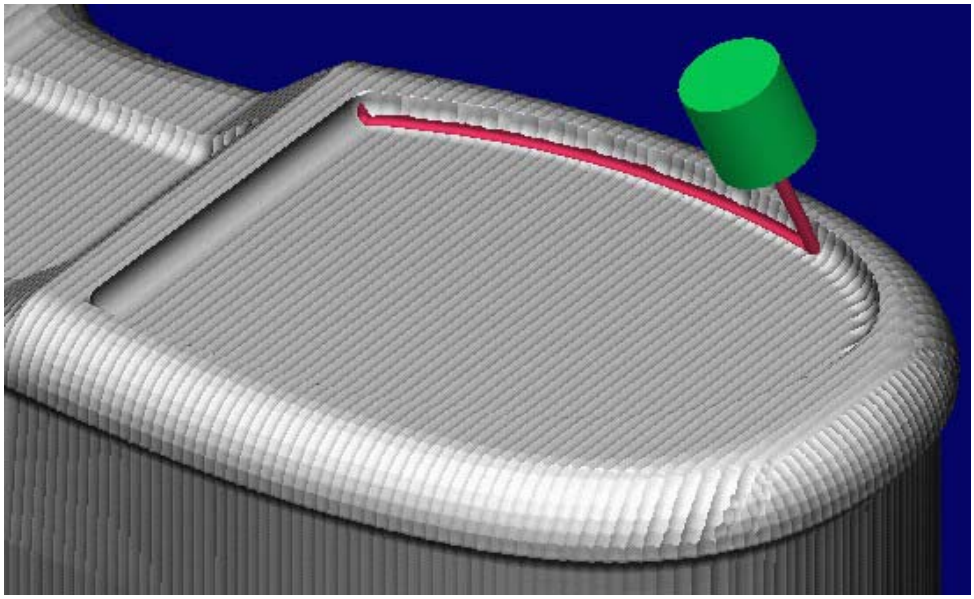


Figure 6.4. Shoe sole model during the 5-axis pencil-cut (top surface)

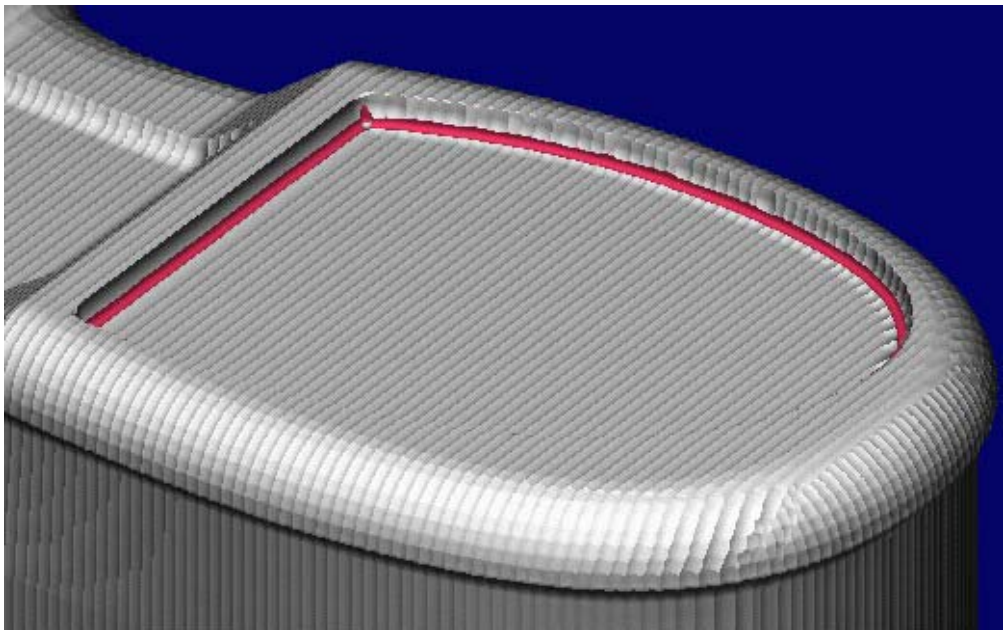


Figure 6.5. Shoe sole model after the 5-axis pencil-cut (top surface)

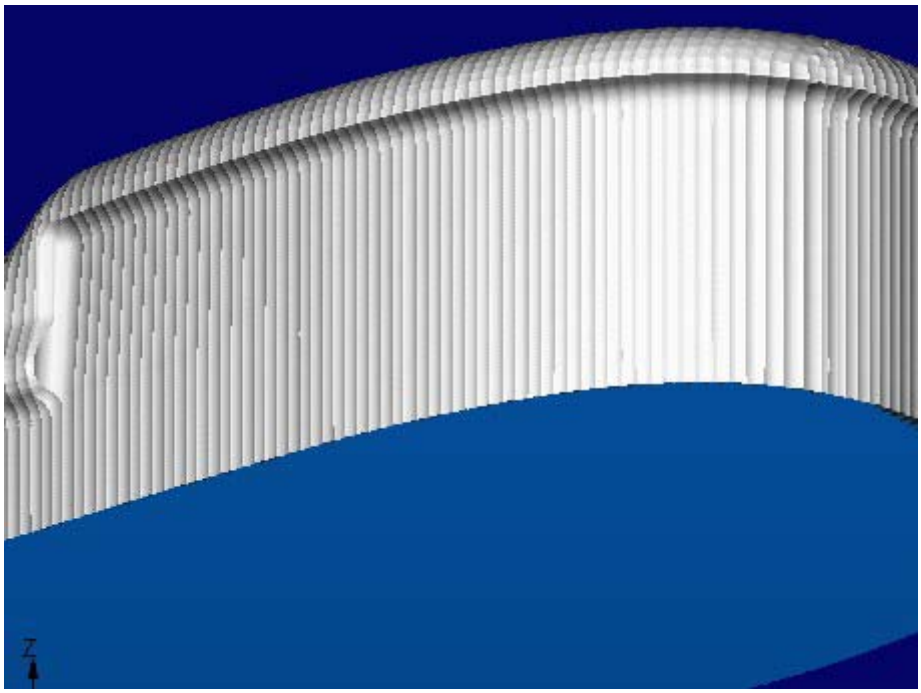


Figure 6.6. The side of Shoe sole model before the 5-axis pencil-cut

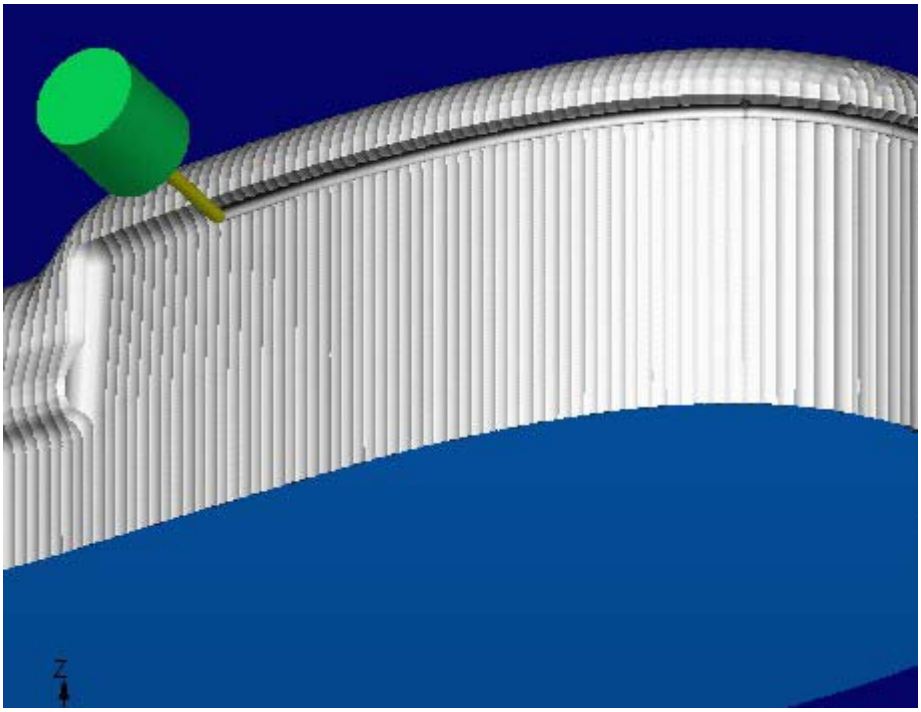


Figure 6.7. The side of Shoe sole model during the 5-axis pencil-cut

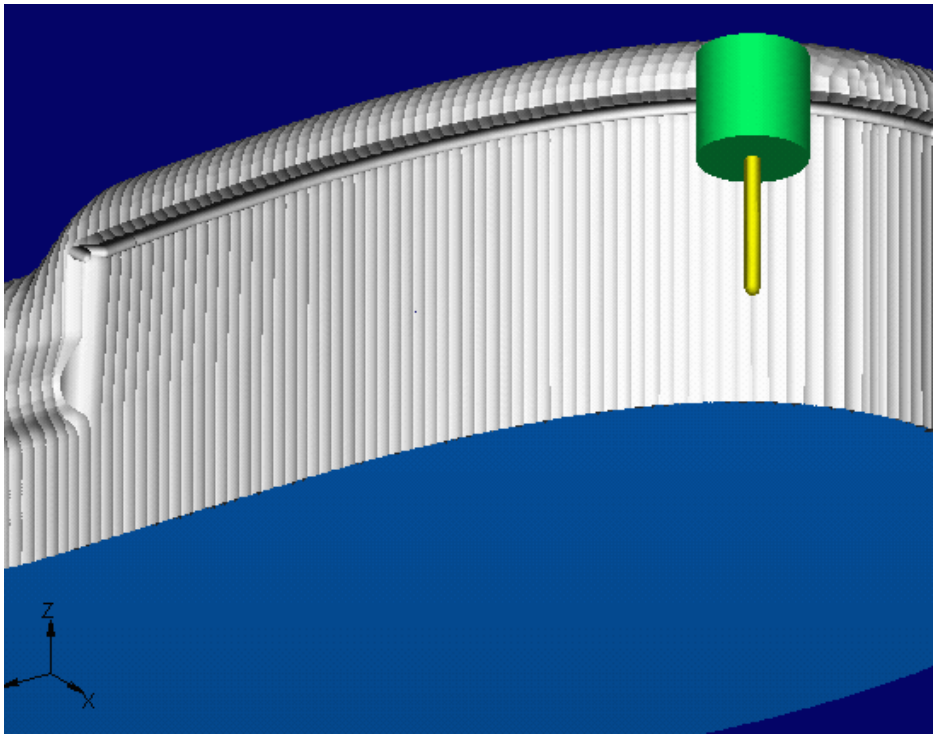


Figure 6.8. The side of shoe sole model after the 5-axis pencil-cut

Figure 6.9 shows a Dexel volume model of a sculptured surface in the haptic virtual sculpting system. Some modification was sculpted on the original sculptured surface as shown in Figure 6.10. The sculpted model was converted into an STL model, as shown in Figure 6.11.

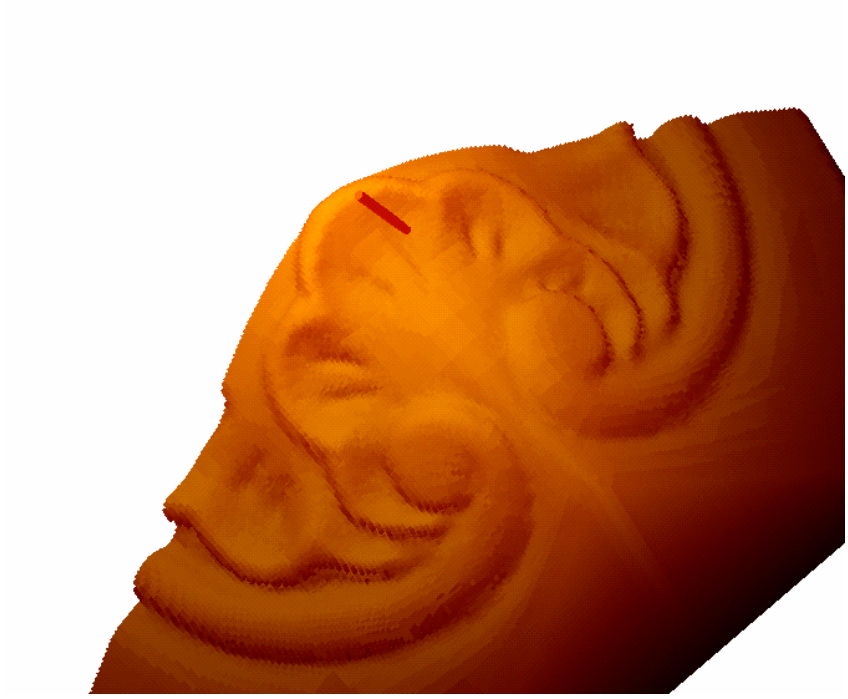


Figure 6.9. A sculptured surface model during the haptic sculpting

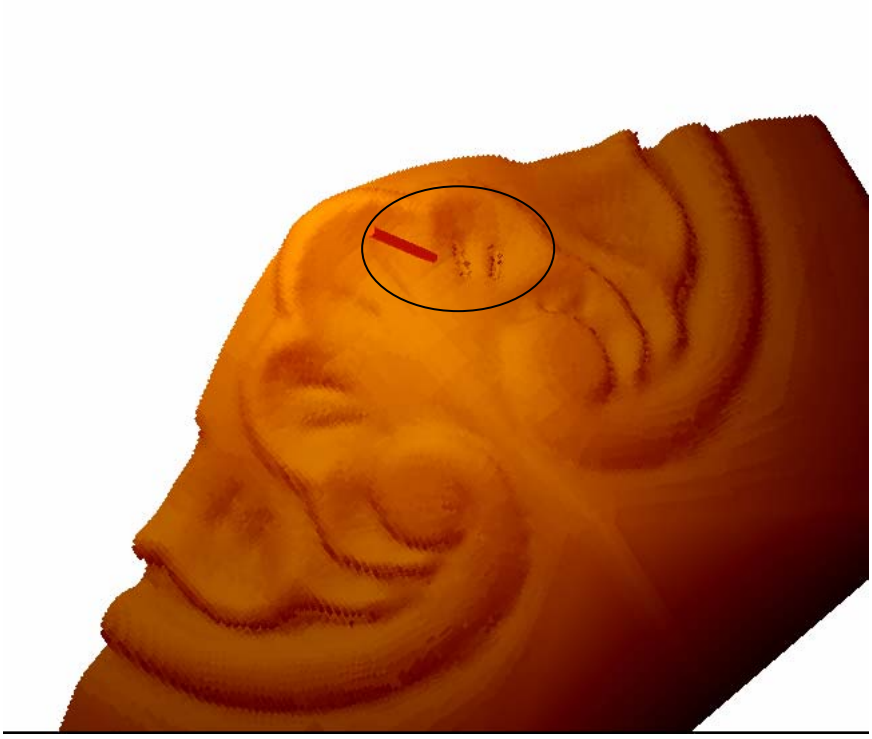


Figure 6.10. Modification on the sculptured surface

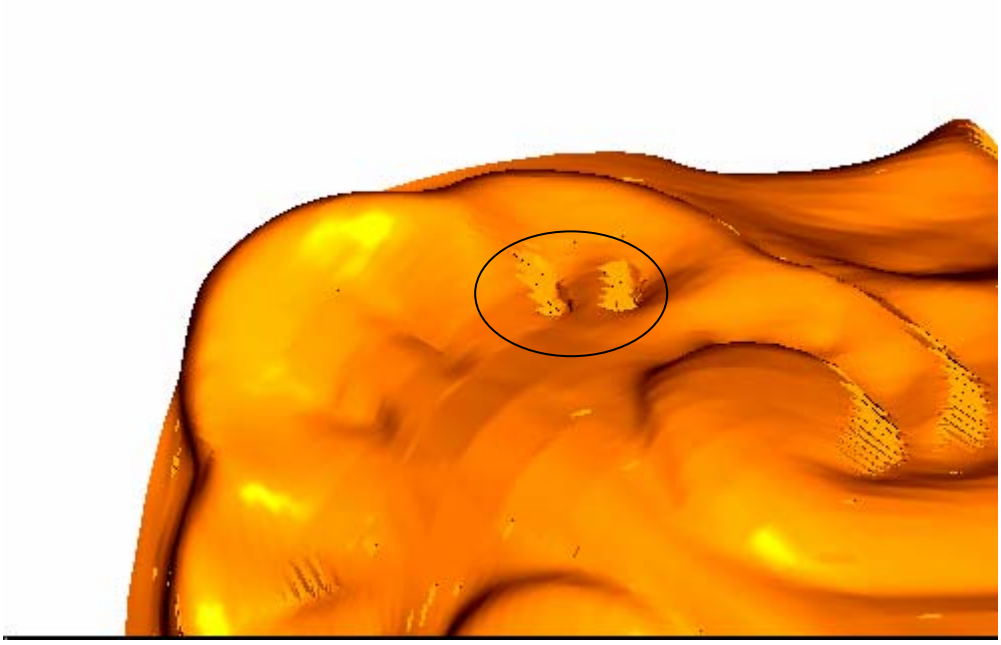


Figure 6.11. The output sculptured surface STL model from the system

Figure 6.12 shows the resultant Dexel volume model from NC simulation of machining a happy Buddha model. This NC simulation software has been developed as part of the virtual sculpting and machining planning system. The happy Buddha model is then converted to STL polyhedral surface model, as shown in Figure 6.13. Some details of the Buddha are shown in Figure 6.14.

Figure 6.12. The NC simulation result for a freeform surface: a happy Buddha Dexel volume model

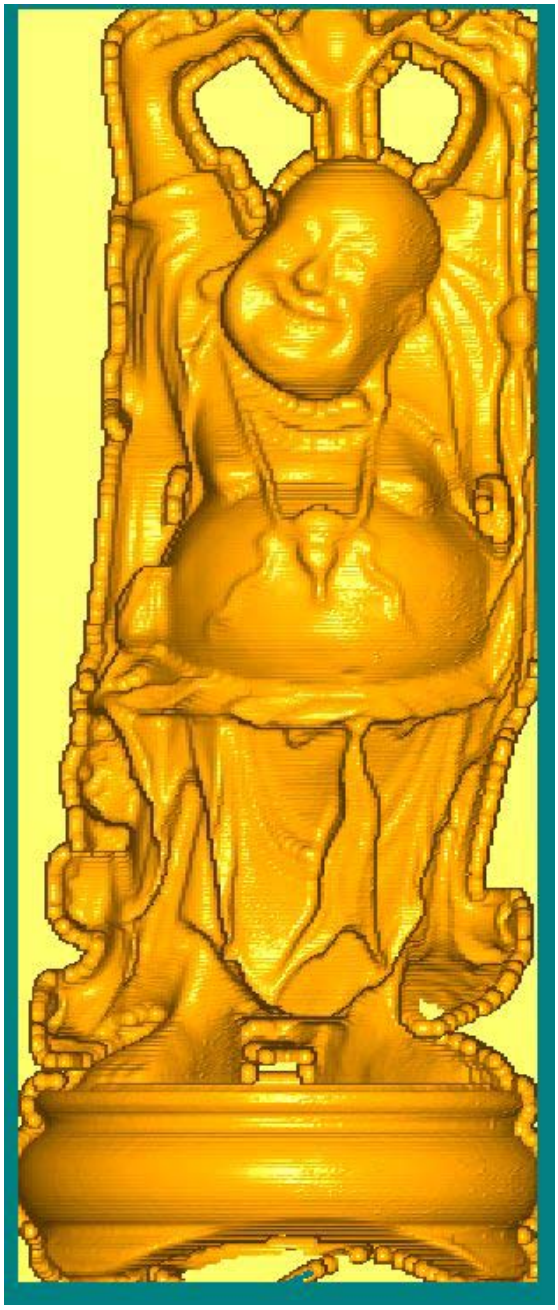


Figure 6.13. The resultant simulation Dexel model has been converted to an STL model (happy Buddha)

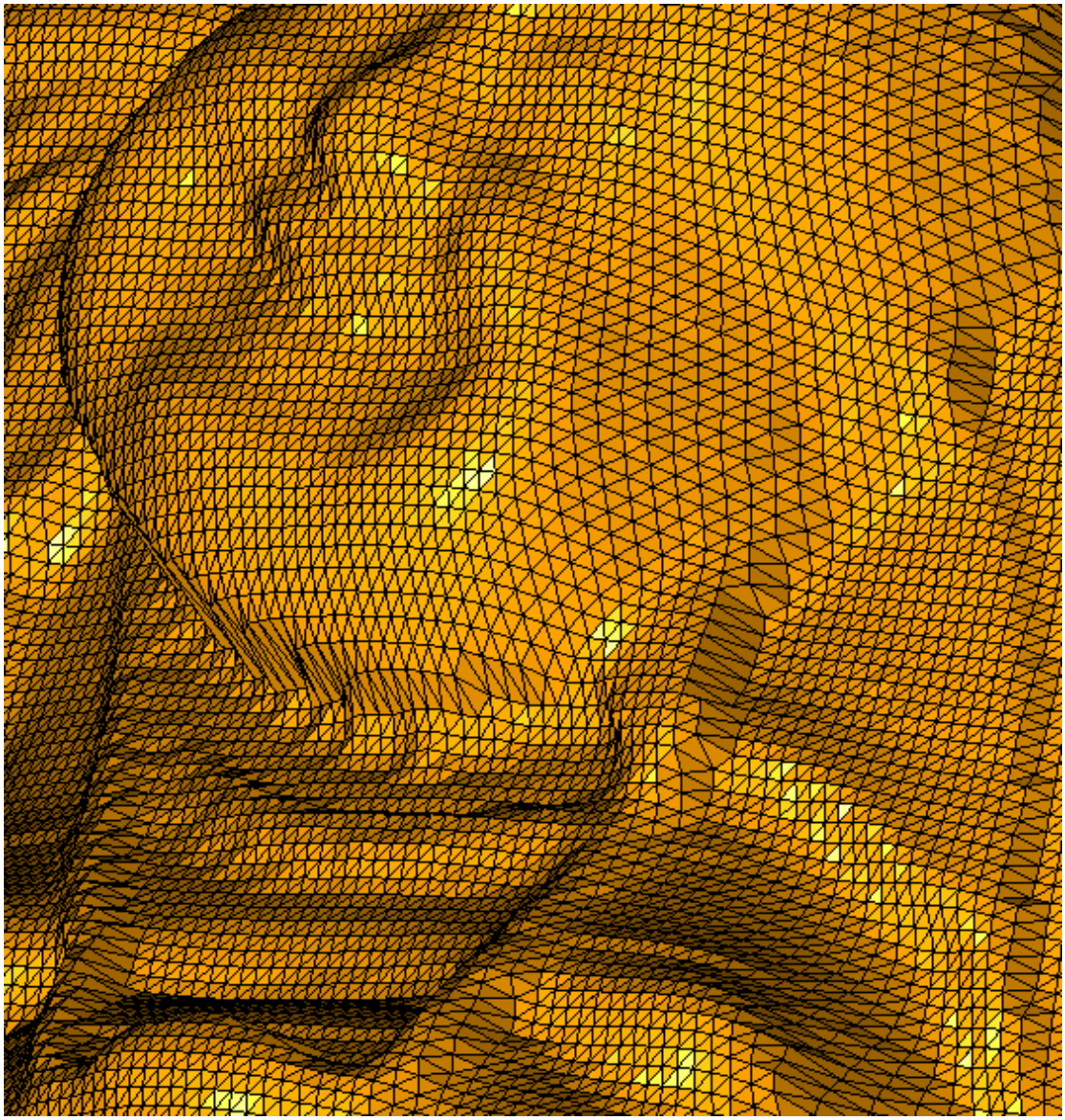
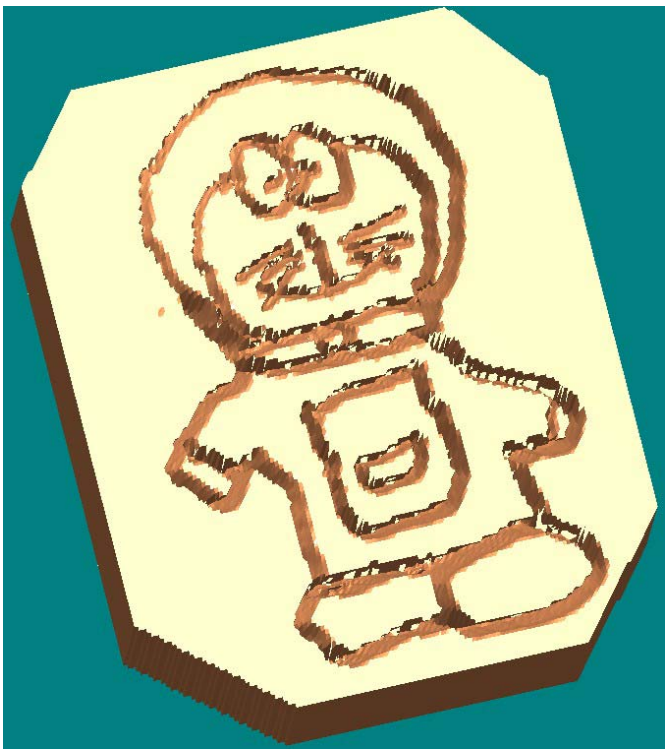


Figure 6.14. The details of the converted STL simulation model: near Buddha's face

Figure 6.15(a) shows a cartoon character (Doraemon) being sculpted in the virtual sculpting system. Figure 6.15(b) shows the output STL model of the cartoon character. Figure 6.16 shows the details of the output STL model near the face.

(a) A cartoon Dexel model during the haptic sculpting



(b) The output cartoon STL model from the system

Figure 6.15. A sculpted cartoon model from the haptic sculpting system

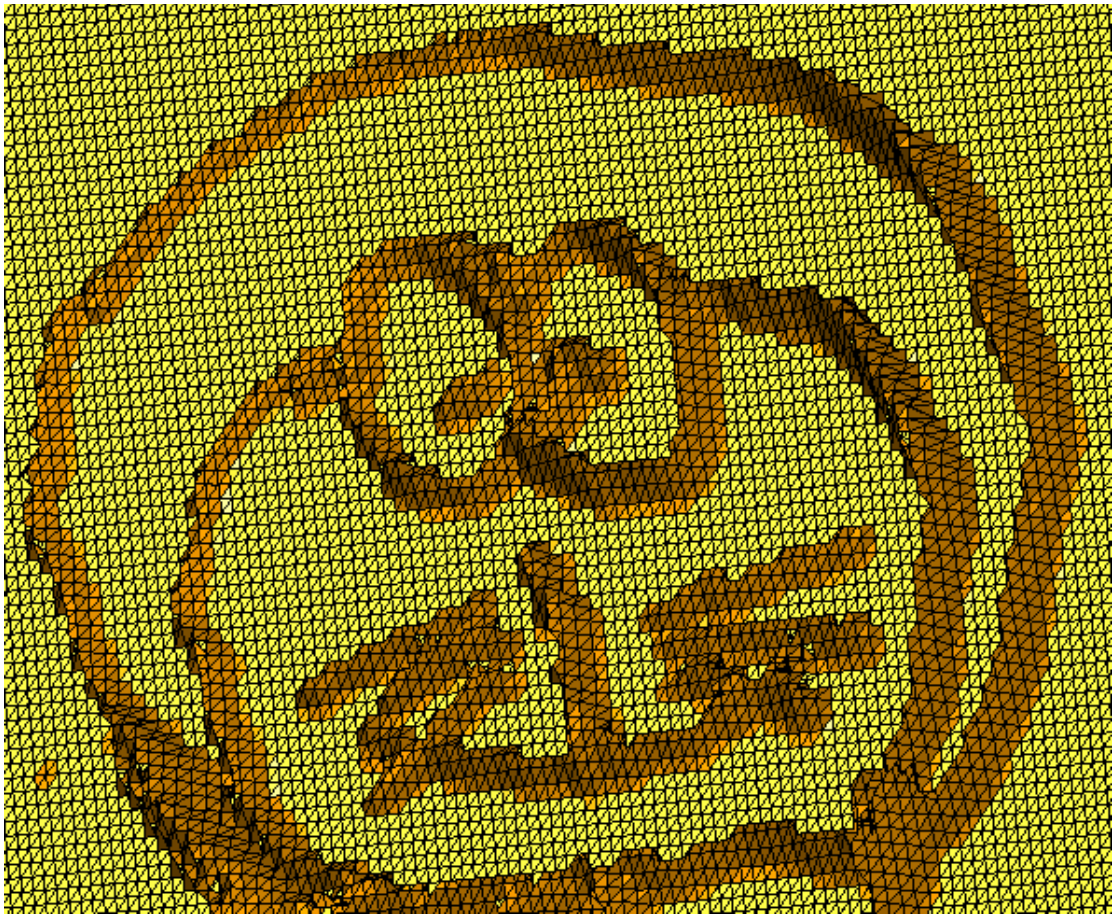
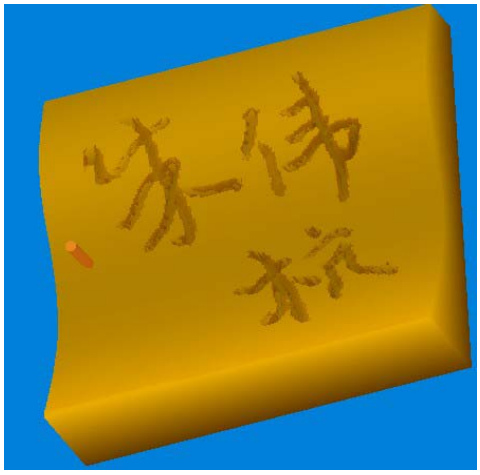
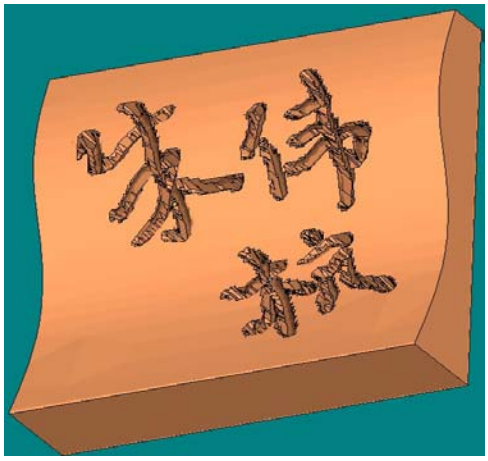


Figure 6.16. Details of the output cartoon STL model near the face

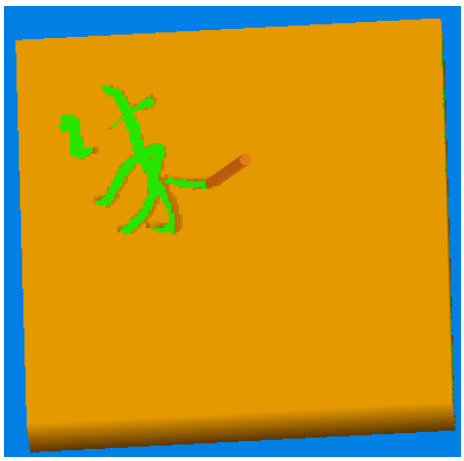
In virtual sculpting system, the output could be either tool motion (output as NC codes) or STL polyhedral surface model. The following simple example illustrates these two kinds of output. Figure 6.17(a) shows some Chinese characters (of the first author's name) being sculpted by the haptic sculpting system. Figure 6.17(b) shows the output STL model of the name. The result implies that the tool can simulate a brush pen very well. The tool motion has been recorded while the name was being written. The tool motion is then output as NC (numerically-controlled) code and played back with a different tool on a thinner stock material, as shown in Figure 6.17(c). The result of NC code playback is shown in Figure 6.17(d).



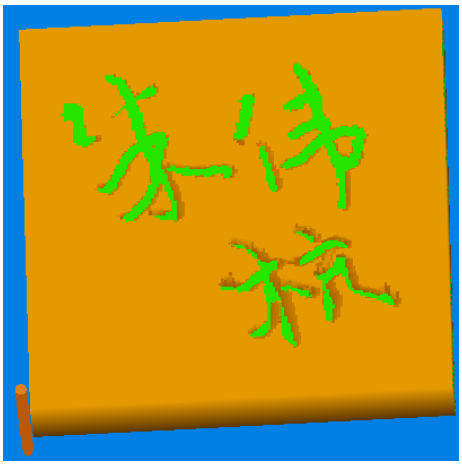
(a) A name Dexel model during the haptic sculpting



(b) The output name STL model from the system



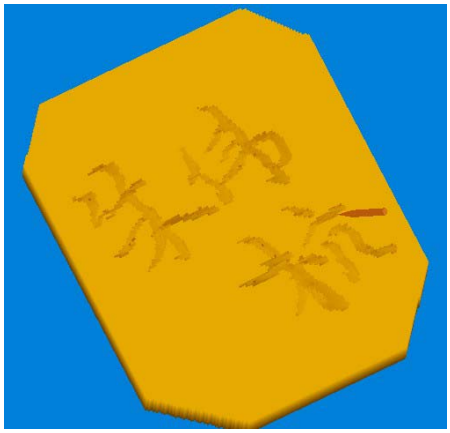
(c) The sculpt motion has been recorded as NC code and is being executed on a different thinner stock material



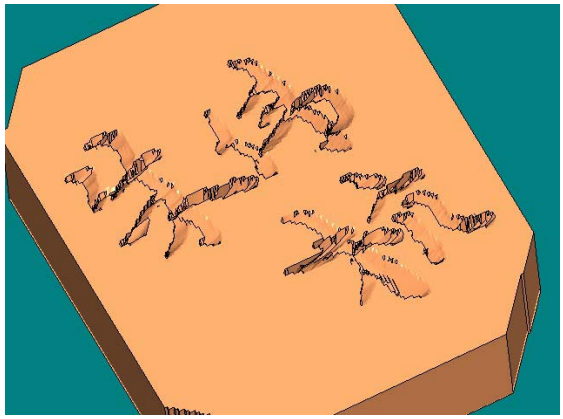
(d) The NC code is executed, using a different tool on a thin board

Figure 6.17. Some Chinese characters (the first author's name) are sculpted by the sculpting system, and the sculpting process is recorded as NC (numerically-controlled) code, and then the NC code is replayed with a different tool and stock material

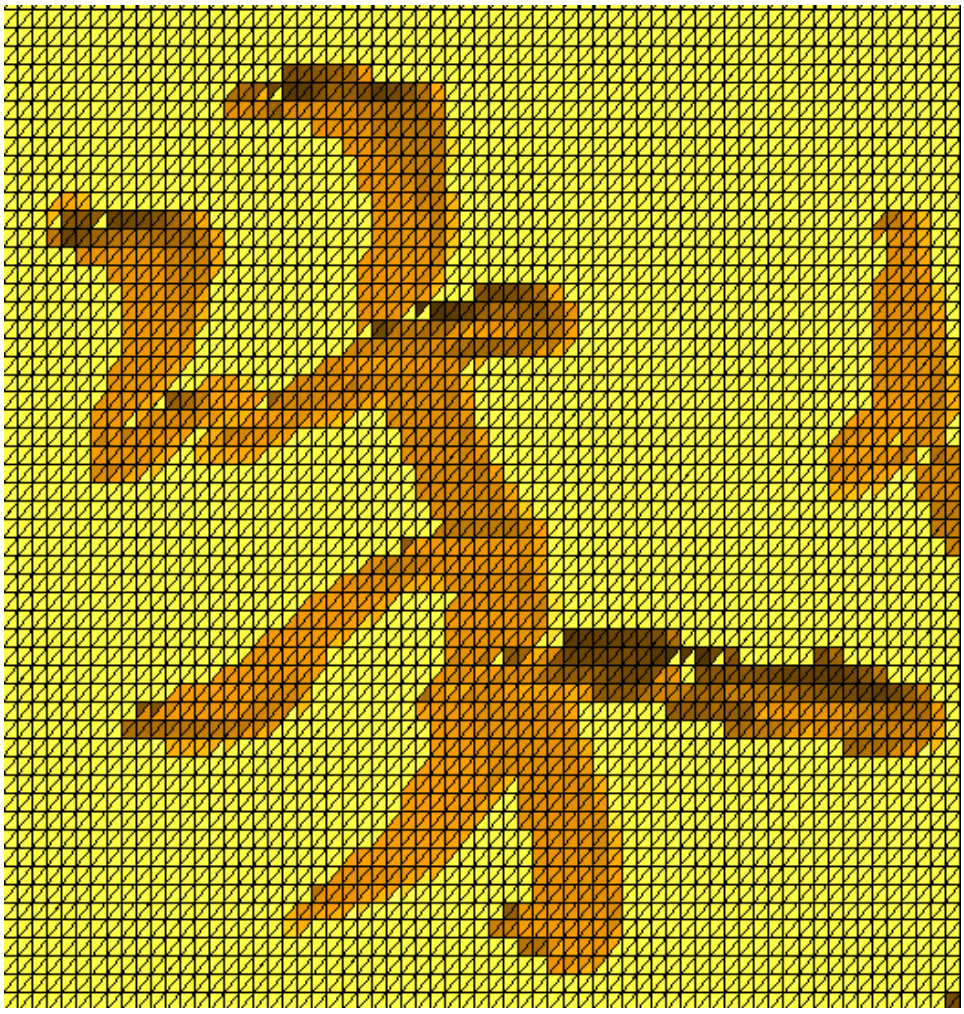
Figure 6.18(a) shows similar Chinese characters sculpted on another stock. Figure 6.18(b) shows the output STL model of the name. The details near one Chinese character are shown in Figure 6.18(c). These intuitive sculpting is very easy to be realized in the developed haptic virtual sculpting system. On the contrary, it is a very difficult to task to realize these designs in a traditional CAD system.



(a) A Dexel model from virtual sculpting: Chinese characters



(b) The Dexel model is converted to an STL model



(c) The detail of the STL model

Figure 6.18. Conversion of a Chinese character Dexel model of Virtual Sculpting to an STL model

6.2 Summary

In this chapter, more interesting implementation examples from the virtual sculpting and machining planning system are presented in addition to the examples shown in the previous chapters. Creative designs can be generated from haptic virtual sculpting system intuitively. These designs are usually difficult to be generated from traditional CAD systems. The designed part can be output as STL polyhedral surface model for further processing in CAD/CAM and RP (Rapid Prototyping) systems. The tool motion of the design process can be recorded as NC commands and played back later. The recorded NC commands can be used for directly reproduction of the creative design, or be used as good suggestions for machining planning. The machining planning system also shows the potential ability that traditional CAM system does not possess. Novel machining strategies can be developed based on interactive haptic interface system. The developed haptic virtual sculpting and machining planning system has revealed its great potentialities in the CAD/CAM field.

Chapter 7
Conclusion and Future Research

The main objective of this research is to develop the novel methodology for a virtual sculpting and machining planning system with a 5-DOF haptic interface. The research involves a novel haptic interface, which provides an intuitive interface between a human user and a computer system. With the involvement of the cutting-edge technology, this research leads to a new virtual sculpting and machining planning system which is promising not only for research but also for commercialization later.

7.1 Conclusions

In this research, the methodology for an integrated haptic sculpting and machining planning system for virtual prototyping and manufacturing has been presented. In the system, a user can virtually sculpt a stock to get an intuitive design by using a haptic interface. The designed model is then processed by the machining planning (manufacturing) module of the system. Haptic interface is also involved in the machining planning to help determine the tool orientation in 5-axis NC (numerically-controlled) tool path generation. A 5-DOF (degree of freedom) haptic device is utilized in the current haptic sculpting and machining planning system. The contributions of this research are summarized as follows:

- ✧ Based on proposed novel methodology, a *dexel-based haptic virtual prototyping* system and a triangulated surface-based machining planning (manufacturing) system have been developed. The integrated system proved to be effective in virtual prototyping and manufacturing.
- ✧ *Dexel-based collision detection* and *force-torque feedback algorithm* have been developed for virtual prototyping module. An OBB (object bounding box)-tree and point-cloud-based two-phase collision detection and force-torque feedback

algorithm are proposed for virtual manufacturing module. These methods contribute to the haptic rendering in the CAD/CAM applications.

- ✧ The output of the virtual sculpting system is STL model (triangular mesh). The in-process model of the virtual sculpting system is Dexel volume model. Hence, algorithms have been proposed for the conversion between STL surface models and Dexel volume models. This bridges the gap between the *dexel-based virtual sculpting system* and *triangular surface-based machining planning system*.

Lab implementation and practical examples have been presented for the *virtual sculpting* and *machining planning* system with a *5-DOF haptic interface*. The techniques presented in this dissertation can lead to a more practical and integrated intuitive design and manufacturing system in the near future. It is also our hope that this research can fuel more interesting haptic research and application in the CAD/CAM area.

7.2 Future Research

The developed haptic sculpting and machining planning system still requires a lot work to be improved in algorithms optimization, visualization, system integration, *etc*. We also look forward to the development of the following techniques: computer speed, faster and robust collision detection algorithms and the design and manufacturing of haptic devices *per se*. The following is a task list that may be considered for improvement and extension of the current system:

- ✧ High speed computing algorithms for real-time tool swept volume generation / intersection. Besides the rotational-symmetrical tools, asymmetrical tools such as chisels may be considered. This will work better if the current 5-DOF haptic interface can be upgraded to a 6-DOF haptic interface, through which the torque around the Z-axis can be fed back to a human user.
- ✧ Visualization algorithms of interactive display for high density dexel model sculpting. Currently the system is limited by the graphics rendering capability of

the video adapter. This requires the development of high speed software algorithms to compensate the limitation.

- ✧ Quick collision detection and response algorithms for different scenarios. In this research, two algorithms are proposed for haptic rendering. However, they cannot cover all kinds of scenarios in CAD/CAM applications. The tool shape is limited to be rotational-symmetrical.
- ✧ A more user-friendly user interface needs to be developed for the integrated system. More modules for different machining strategies need to be developed to facilitate different kinds of NC machining planning.
- ✧ The dexel representation of sculpted objects is not good in surface finish in vertical surfaces. Hence a triple-ray representation may be used to enhance the surface quality. This triple-ray representation is like 3 Dexel models overlapped together in 3 different directions, namely X, Y and Z axes.
- ✧ The current dexel-based virtual sculpting system may be combined with spline-based haptic sculpting system. This will assimilate the advantages of both systems.
- ✧ Haptic tracing with NURBS surface may be considered. NURBS is a universal representation in CAD/CAM systems. Current technique only uses a point to trace NURBS [Thompson 1997]. A virtual tool defined with implicit functions may be considered to trace the NURBS with 5 or 6-DOF haptic interface. The internal representation of NURBS may be transformed into NURP (Non-Uniform Rational Polynomial) surface.
- ✧ The stability of the haptic interface is also an important problem. Haptic controller, both hardware and software, needs to be designed carefully so that the haptic interface is stable during the interaction process. Currently most stability research is done on 1-DOF haptic interface. More work is need on more DOF haptic interface.

This task list is by no means a complete list. It shows that haptics research may have good expansion both in span and in depth. It is expected that haptics research will continue to boom in various fields, including CAD/CAM.

References

[ACM SM 2001] Haptic Rendering in Virtual Environments with applications to CAD and Manufacturing, ACM Solid Modeling ’01 Tutorial T2, 2001

[Abdel-Malek 2003] Abdel-Malek, K., Blackmore, D., and Joy, K., (submitted), Swept volumes: foundations, perspectives, and applications, *International Journal of Shape Modeling*

[Adachi 1995] Adachi Y., Kumano T., and Ogino K., Intermediate representations for stiff virtual objects, *Proc. IEEE Virtual Reality Annual Intl. Symposium* (1995) 203-210

[Adachi 2003] Adachi, Y., Personal communication, 2001-2003

[Balasubra 2002] Balasubramaniam M, Ho Stephen, Sarma S, Adachi Y, Generation of collision-free 5-axis tool paths using a haptic surface, *Computer-Aided Design* 34 (2002) 267-279

[Baraff 1994] Baraff David, Fast contact force computation for non-penetrating rigid bodies, *SIGGRAPH* 94, 1994

[[Baraff 1997] Baraff David, An introduction to physically based modeling: Rigid body simulation, *SIGGRAPH* 97 Course Notes, 1997

[Barequet 1998] G. Barequet, C.A. Duncan, and S. Kumar, “RSVP: A Geometric Toolkit for Controlled Repair of Solid Models”, *IEEE Transactions on Visualization and Computer Graphics*, 4(2), (1998), 162-177

[Basdogan 2001] Basdogan C., Real-time simulation of dynamically deformable finite element models using modal analysis and spectral Lanczos decomposition methods, *Proceedings of the Medicine Meets Virtual Reality (MMVR’2001) Conference*, 2001

[Basdogan 2002] Basdogan C. and Srinivasan M. A. Haptic Rendering in Virtual Environments, in *Handbook of Virtual Environments*, edited by Kay Stanney, Chapter 6, Lawrence Earlbaum Inc., London, 2002

[Bastogan Web] Bastogan, C., http://network.ku.edu.tr/~cbasdogan/Tutorials/sig9902.pdf

[Benouamer 1997] Benouamer M.O. and Michelucci D., Bridging the gap between CSG and Brep via a Triple Ray Representation, *Proceedings of the fourth ACM symposium on Solid modeling and applications*, May 14-16, 1997, 68-79

[Biggs 2002] Biggs, J. and Srinivasan M. A. Haptic Interfaces, in *Handbook of Virtual Environments*, edited by Kay Stanney, Chapter 5, Lawrence Earlbaum, Inc., London, 2002

[Blackmore 1997] Blackmore D, Leu M.C. and Wang L.P, The Sweep-Envelope Differential Equation Algorithm and its Application to NC

Machining Verification, *Computer-Aided Design*, 29 (1997) 629-637

[Burdea 1996] Burdea Grigore C., *Force and Touch Feedback for Virtual Reality*, New York, John Wiley & Sons, Inc, 1996

[Chiou 1999] Chiou C.-J. and Lee. Y.S, A shape-generating approach for multi-axis machining G-buffer models, *Computer-Aided Design*, 31 (1999) 761-776

[Chiou 2002] Chiou C.-J. and Lee. Y.-S, A machining potential field approach to tool path generation for multi-axis sculptured surface machining, *Computer-Aided Design*, 34 (2002) 357-371

[Choi 1997] Choi B.K. Kim D.H. and Jerard R.B., C-space approach to tool-path generation for die and mould machining, *Computer-Aided Design*, 1997, Vol. 29, No. 9, 657-669.

[Choi 1998] Choi B.K. and Jerard R.B., Sculptured surface machining, Boston, Kluwer Academic Publishers, 1998

[Choi 2003] Choi S.H. and Chan A.M.M., A Virtual Prototyping System for Rapid Product Development, *Computer Aided-Design*, 2003, Article in Press

[Coquillart 1990] Coquillart, S., Extended Free-Form Deformation: a Sculpting Tool for 3D Geometric Modeling, *Computer Graphics* (*Proc. SIGGRAPH '90*), Vol. 24, No. 4, 187~196

[Craig 1989] Craig J.J. Introduction to Robotics: Mechanics and Control, New York, Addison-Wesley Publishing Company, Inc, 1989

[Dachille 2001] Dachille IX., F., Qin, H. and Kaufman, A., A Novel Haptic-based Interface and Sculpting System for Physics-based Geometric Design, *Computer-Aided Design*, 33 (2001) 403-420

[DELMIA Web] http://www.delmia.com/

[Fang 1998] Fang Y C, Hsieh CC, Kim M J, Chang J J, Woo T C, Real time motion fairing with unit quaternions, *Computer-Aided Design* 30 (1998) 191-198

[Farin 1996] Farin, G., Curves and Surfaces for Computer Aided Geometric Design**,** Academic Press, San Diego, CA, 4th edition

[Gaylean 1991] Gaylean T. A. and Hughes J. F., Sculpting: An Interactive Volumetric Modeling Technique, *Computer Graphics* (*Proc. SIGGRAPH '91*), Vol. 25, No. 4, 267-274

[Gelder 1994] Gelder A.V., Wilhelms, J., Topological Considerationsin Isosurface Generation, ACM Transaction on Graphics, Vol. 13, No. 4, October 1994, 337-375

[Glaeser 1999] Glaeser G., Wallner J., Pottmann H., Collision free 3-axis milling and selection of cutting tools, *Computer-Aided Design* 31 (1999) 225-232

[Gottschalk 1996] Gottschalk S., Lin M.C., and Manocha D., OBBTree: A Hierarchical Structure for Rapid Interference Detection, *Computer Graphics (Proc. SIGGRAPH'96)*, August, 1996,

171-180

[Gray 2002] Gray P., Bedi S. and Ismail F., Rolling ball method for 5-axis surface machining, *Computer-Aided Design*, 2002, Article in press

[Gregory 2000] Gregory Arthur, Mascarenhas A., Ehmann S., Lin M. and Manocha D., Six Degree-of-freedom haptic display of polygonal models, http://www.cs.unc.edu/~geom/6DHaptics

[Hamilton 1969] Hamilton W., *Elements of Quaternions*, Vol. I-II. Chelsea Publishing Company, Chelsea, 1969

[Held 1997] Held, M., ERIT: A collection of efficient and reliable intersection tests. *Journal of Graphics Tools*, 2(4):25-44, 1997

[Ho 1997] Ho Chih-Hao, Basdogan C. and Srinivasan M.A., Haptic Rendering: Point- and Ray-Based Interactions, at the second *PHANToM User's Group Workshop*, 1997

[Ho 2001] Ho S., Sarma S., Adachi Y., Realtime interference analysis between a tool and an environment, *Computer-Aided Design* 33 (2001) 935-947

[Huang 1994] Huang Y., Oliver J.H., NC Milling Assessment and Tool Path Correction, *Computer Graphics (Proc. SIGGRAPH '94)*, July 24-19, 1994, 287--294,

[Hwang 1998] Hwang J.S. and Chang T.C., Three-axis machining of compound surfaces using flat and filleted endmills, *Computer-Aided Design*, 1998, Vol. 30, No. 8, 641-647

[Hubbard 1996] Hubbard P.M., Approximating polyhedra with spheres for time-critical collision detection, *ACM Transactions on Graphics*, Vol. 15, No. 3, July 1996, pp179-210

[Jensen 2002] Jensen C.G., Red W.E, Pi, J., Tool selection for five-axis curvature matched machining, *Computer-Aided Design*, 34 (2002) 251-266

[Jun 2003] Jun C.-S., Cha K and Lee Y.S., Optimizing tool orientation for 5-axis machining by configuration-space search method, *Computer-Aided Design*, 35 (2003) 549-566

[Kaufman 1993] Kaufman A., Cohen D., and Yagel R., Volume Graphics, *IEEE Computer*, 26, 7 (July 1993), 51-64

[Klosowski 1998] Klosowski J.T., Held M., Mitchell J.S.B., Sowizral H., Zikan K., Efficient Collision Detection Using Bounding Volume Hierarchies of k-DOPs, *IEEE Transactions on Visualization and Computer Graphics*, Vol. 4, No. 1, 1998, 21-36

[Koc 2000] Koc K., Ma Y., and Lee Y-S., Smoothing STL Files by Max-Fit Biarcs Curves for Rapid Prototyping, *Rapid Prototyping Journal*, 6(3) (2000) 186-204

[Konig Web] Konig A.H. and Groller E., Real Time Simulation and Visualization of NC milling Processes for Inhomogeneous Materials on Low-End Graphics Hardware,

http://www.cg.tuwien.ac.at/home/

[Lee 1995] Lee Y.S, Two-Phase approach to global tool interference avoidance in 5-axis machining, *Computer-Aided Design*, 27 (1995), 715-729

[Lee 1998] Lee Y.S, Mathematical modeling using different endmills and tool placement problems for 4- and 5-axis NC complex surface machining, *International Journal of Production Research*, 1998, Vol. 36, No.3, 785-814

[Lee 2000] Lee Y.S., Ma Y., and Jegadesh G., Rolling-Ball Method and contour marching approach to identifying critical regions for complex surface machining, *Computer In Industry*, Vol. 41, No. 2, 2000, 163-180

[Leu 2002] Leu M.C., Velivelli A. Peng X., Creating freeform model by carving virtual workpiece with haptic interface, *Proceedings of ASME IMECE (Intl. Mech. Engineering Congress and Exposition)* Nov. 17-22, 2002, New Orleans, Louisiana, 2002-32467

[Lorensen 1987] Lorensen W.E., Cline H.E., Marching Cubes: A High Resolution 3D Surface Construction Algorithm, *Computer Graphics*, Vol. 21, 4, (July 1987), 163-169

[McNeely 1999] McNeely W.A., Puterbaugh K.D., Troy J.J, Six degree-of-freedom rendering using voxel sampling, *SIGGRAPH 99*, 401-408

[Mann 2002] Mann S. and Bedi S., Generalization of the imprint method to general surfaces of revolution for NC machining, *Computer-Aided Design*, 34 (2002) 373-378

[Meyers 1992] Meyers D., Skinner S., and Sloan K., Surfaces from Contours, *ACM Transaction on Graphics*, 11, 3, July 1992, 228-258

[Nooduddin 2003] Nooruddin F.S. and Turk G., Simplification and repair of polygonal models using volumetric techniques, *IEEE Transactions on Visualization and Computer Graphics*, Vol. 0, No. 2, April-June 2003, 191-205

[O'Connell 1993] O'Connell JM and Jablokow AG., Construction of solid models from NC machining programs, *Proceedings of ASME on Manufacturing Science and Engineering 1993*; PED64: 157-166

[Qin 1996] Qin Hong and Terzopoupos D., D-NURBS: A Physics-Based Framework for Geometric Design, *IEEE Transactions on Visualization and Computer Graphics*, Vol. 2, No. 1, March 1996

[Ren 2002] Ren Y., Lee Y.S. and Yau H.T., Contraction tool method and clean-up tool path generation for machining polynomial models, (in revision), submitted to *Computer In Industry*, 2002

[Roy 1999] Roy U and Xu Y., Computation of a geometric modle of a machined part from its NC machining programs, *Computer Aided-Design*, 31 (1999) 401-411

[Schroeder 1992] Schroeder W.J., Zarge J.A. and Lorensen W.E., Decimation of Triangle Meshes, *Computer Graphics (Proc. SIGGRAPH '92)*, Vol. 26, No. 2, July 1992, 65--70

[Srinivasan 1997] Srinivasan, Mandayam A. and Basdogan, Cagatay, Haptics in virtual environments: taxonomy, research status, and challenges, *Computer & Graphics*, Vol. 21, No. 4, 1997, 393-404

[Thompson 1997] Thompson II T.V., Johnson D.E. and Cohen E., Direct Haptic Rendering of Sculptured Models, *Proceedings Symposium on Interactive 3D Graphics*, Providence, RI, April 27-30, 1997, 1-10

[Van Hook 1986] Van Hook T., Real-Time Shaded NC Milling Display, *Computer Graphics (Proc. SIGGRAPH '86)*, Vol. 20, No. 4 (August 1986), 15-20

[Wang 1995] Wang W. S. and Kaufman A.E., Volume sculpting, *ACM Symposium on Interactive 3D Graphics*, Monterey CA USA, 1995, 151-156 and 214

[Yoon 2001] Yoon J.-H, Pottmann H. and Lee Y.-S, Locally optimal cutting positions for 5-axis sculptured surface machining, *Computer-Aided Design*, 2001, Article in press

[Zhu 2003a] Zhu W. and Lee Y.S., Haptic sculpting and machining planning with 5-DOF Haptic Interface for Virtual Prototyping and Manufacturing, Accepted to appear in the *International Conference on Advanced Research in Virtual and Rapid Prototyping (VRAP)*, Leiria, Portugal, Oct, 2003

[Zhu 2003b] Zhu W. and Lee Y.S., Haptic sculpting and pencil-cut planning in virtual prototyping and manufacturing, Accepted to appear in the *ASME IMECE (International Mechanical Engineering Congress and Exposition) Conference*, Nov, 2003

[Zhu 2003c] Zhu W. and Lee Y.S., Analytical methodology of updating Dexel volume model for virtual sculpting with a 5-DOF haptic interface, (working paper), 2003

[Zhu 2003d] Zhu W. and Lee Y.S., Five-axis pencil-cut machining planning and virtual prototyping with a 5-DOF haptic interface, (working paper), 2003

[Zhu 2003e] Zhu W. and Lee Y.S., A marching algorithm of constructing polyhedral models from Dexel models for haptic virtual sculpting (working paper), 2003

[Zilles 1995] Zilles C.B., and Salisbury J.K., A constraint-based god-object method for haptics display, *Proc. IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, Pittsburgh, PA (1995) 146-151

[Zorin 2000] Zorin Denis, Schroder P., Subdivision for Modeling and Animation, *SIGGRAPH* 2000 course notes, 2000

Appendices

Appendix A: Jacobian Matrix for haptic device

This appendix is mainly about the relation between the haptic device motor torques and the generated torque and force on the user hand [Adachi 2003]. First the mathematical fundamental of the Jacobian Matrix is presented. Then it is used in deriving the relation between the motor torques and the resultant torque and force.

1. Foundations

Suppose that the following relation holds between a *k*-dimensional vector $\xi = [\xi_1 \quad \xi_2 \quad \cdots \quad \xi_4]^T$ and an *l*-dimensional vector $\eta = [\eta_1 \quad \eta_2 \quad \cdots \quad \eta_4]^T$:

$$\eta_j = f_j(\xi_1 \quad \xi_2 \quad \cdots \quad \xi_k), j = 1,2,\cdots,l \tag{A-1}$$

Then the *l* * *k* matrix

$$J_\eta(\xi) = \begin{bmatrix} \frac{\partial \eta_1}{\partial \xi_1} & \frac{\partial \eta_1}{\partial \xi_2} & \cdots & \frac{\partial \eta_1}{\partial \xi_k} \\ \frac{\partial \eta_2}{\partial \xi_1} & \frac{\partial \eta_2}{\partial \xi_2} & \cdots & \frac{\partial \eta_2}{\partial \xi_k} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial \eta_l}{\partial \xi_1} & \frac{\partial \eta_l}{\partial \xi_2} & \cdots & \frac{\partial \eta_l}{\partial \xi_k} \end{bmatrix} \hat{=} \frac{\partial \eta}{\partial \xi^T} \tag{A-2}$$

is called the Jacobian Matrix of η with respect to ξ. Further, suppose that ξ and η are function of time, then differentiating equation (A-1) with respect to time and substituting equation (A-2) yields

$$\dot{\eta} = J_\eta(\xi)\dot{\xi} \tag{A-3}$$

As is seen from equation (A-3), the Jacobian matrix is an extension of the derivative for a scalar variable to the case of vectors.

2. Haptic device structure

Suppose that the joints of a manipulator of a haptic device are numbered 1, 2, …, n, starting from the base of the manipulator (grip). The displacement of joint i is denoted q_i and is called the joint variable. The collection of joint variables

$$q = \begin{bmatrix} q_1 & q_2 & \cdots & q_n \end{bmatrix}^T \tag{A-4}$$

is called the joint vector. The position of the end-effector of the manipulator is denoted by the m-dimensional vector

$$r = \begin{bmatrix} r_1 & r_2 & \cdots & r_n \end{bmatrix}^T \tag{A-5}$$

where m <= n. For a general case where the end-effector can take an arbitrary position and orientation in three-dimensional Euclidean space, we have m = 6.

The relation between r and q, determined by the manipulator mechanism, generally is non-linear. It is assumed that this relation is given by

$$r = f_r(q) \tag{A-6}$$

Using the Jacobian matrix, one can express the relation between the end-effector velocity and the joint velocity of a manipulator in a compact form. Differentiating equation (A-6) with respect to time yields

$$\dot{r} = J_r(q)\dot{q} \tag{A-7}$$

where $\boldsymbol{J_r(q)}$, the Jacobian matrix of $\boldsymbol{r}$ with respect to $\boldsymbol{q}$, is given by

$$j_r(q) = \frac{\partial r}{\partial q^T} \tag{A-8}$$

The matrix $\boldsymbol{J_r(q)}$ will henceforth be written as $\boldsymbol{J_r}$, for simplicity.

From equation (A-7), the virtual displacement of the end-effector, δr, and the virtual displacement of the joints, δq, satisfies

$$\delta r = J_r \delta q \tag{A-9}$$

On the other hand, from the principle of virtual work we have

$$(\delta q)^T \tau = (\delta d)^T f \tag{A-10}$$

where $\tau = [\tau_1 \quad \tau_2 \quad \cdots \quad \tau_4]^T$ is the joint driving force and $f \in \Re^3$ is the generated force. Hence, from equation (A-9) and (A-10) we obtain

$$\tau = J_r^T f \tag{A-11}$$

This is the relation between the joint driving torque and the generated force.

Appendix B: Some specification of the current haptic device

This appendix records the specification of main components of the current haptic device. These components include motors, encoders, D/A board and counter board. They are of high precision. The specifications of them are as follows,

Part	#	*Specification*
Motor	6	Miniaturized Coreless DC Servomotor @25 Series NC-258102, from *Chiba Precision*, Japan
Encoder for motor	6	ENC-258101 2000/3CH (E-3030-100 (1000P/R) / 30mm diameter) from *Chiba Precision*, Japan
Encoder at probe	1	SE10-200 from *Chiba Precision*, Japan
Amplifier	6	A-001 from *Chiba Precision*, Japan
D/A board	1	PCIDAC08, 12 bit, 8 channels, from *CyberResearch Inc*. USA
Counter board	1	CNT32-8M(PCI), 32 bit, 8 channels, from *CONTEC*, Japan

Table B-1 Haptic device components specification

The color specification for the encoder wires is as follows,

Green	Phase-A
Yellow	Phase-B
White	Phase-Z
Red	+5V
Black	0V
Blue	Not Used

Table B-2 Encoder wires' specification

The connectors are sponsored by JST *Inc*.