A VIRTUAL TEMPORAL BONE DISSECTION SIMULATOR

A Thesis

Presented in Partial Fulfillment of the Requirements for

the Degree Master of Science in the

Graduate School of The Ohio State University

By

Jason Allen Bryan, B.S.

* * * * *

The Ohio State University
2001

Master's Examination Committee:

Dr. Roger Crawfis, Advisor
Dr. Raghu Machiraju
Dr. Han-Wei Shen

Approved by

Adviser
Department of Computer and
Information Science
ABSTRACT

The Temporal Bone Dissection Simulator is an ongoing project funded by a grant from The National Institute on Deafness and other Communication Disorders (NIDCD) to construct a dissection simulator of the regional anatomy of the temporal bone. The goal of this project is to create a virtual performance environment for dissection of temporal bone with a focus on preoperative assessment and surgical planning. The simulator will be used in efficacy studies through integration with an otolaryngology curriculum, with eventual extension to simulation of surgical procedures. Through the use of leading edge visualization technology and by incorporating aspects of an intelligent tutor, the simulation system provides a means for learning temporal bone anatomy and surgical procedures while maintaining a surgical context. This document presents a summary of the current state of the project. The temporal bone is identified, and motivation for a virtual dissection simulation provided. Parallel research is described, and the approach of this project presented. Three techniques that address specific problems encountered in the development of the dissection simulator are introduced. The first technique provides a method for rendering a five channel segmented volume while interactively modifying the visibility of each segment. This technique is necessary for aspects of the incorporated intelligent tutor. The second technique provides a means for
improving volume rendering frame rates during local modification. This technique solves the problem of maintaining sufficient interactivity while a virtual dissection is in progress. The third technique supplies an extra modality to volume interaction by providing simple haptic (force) feedback of volumetric data. This technique is required in order to increase realism and provide an appropriate tactile response to the user. The current status of the project is presented, along with preliminary user feedback and future goals.
ACKNOWLEDGMENTS

I wish to thank Don Stredney for providing the encouragement, opportunity, and reward for undertaking the rigors of graduate school, and for his assistance on this thesis.

I also thank my advisor, Roger Crawfis, for his tremendous contribution in helping me write this thesis and his guidance throughout my graduate experience.

I am grateful to my best friend, Kelly Summers, for her unwavering love and support throughout my college career, and to my parents for their encouragement and confidence in me.
VITA

October 28, 1976...................................................... Born – Columbus, Ohio

1999................................................................. B.S. Computer Science and Engineering,
The Ohio State University

1998-2001............................................................ Undergraduate Research Assistant,
Ohio Supercomputer Center

2001-present......................................................... Director of Software Development,
Interface Lab,
Ohio Supercomputer Center

PUBLICATIONS

Research Publication


FIELDS OF STUDY

Major Field: Computer and Information Science: Computer Graphics
# TABLE OF CONTENTS

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Abstract</td>
<td>ii</td>
</tr>
<tr>
<td>Acknowledgments</td>
<td>iv</td>
</tr>
<tr>
<td>Vita</td>
<td>v</td>
</tr>
<tr>
<td>List of Tables</td>
<td>ix</td>
</tr>
<tr>
<td>List of Figures</td>
<td>x</td>
</tr>
<tr>
<td><strong>Chapters:</strong></td>
<td></td>
</tr>
<tr>
<td>1. Introduction</td>
<td>1</td>
</tr>
<tr>
<td>1.1. Background and Existing Methods</td>
<td>1</td>
</tr>
<tr>
<td>1.2. Previous Work on Temporal Bone Simulations</td>
<td>4</td>
</tr>
<tr>
<td>1.3. Approach to the Construction of this Simulator</td>
<td>5</td>
</tr>
<tr>
<td>1.3.1. Key Components to the Effective Simulator</td>
<td>5</td>
</tr>
<tr>
<td>1.3.2. Requirements of the Simulator System</td>
<td>6</td>
</tr>
<tr>
<td>1.3.3. System Hardware</td>
<td>8</td>
</tr>
<tr>
<td>1.3.4. Data Used by the Simulator</td>
<td>8</td>
</tr>
<tr>
<td>1.3.5. Visualization Methods</td>
<td>11</td>
</tr>
<tr>
<td>1.4. Thesis Statement</td>
<td>12</td>
</tr>
<tr>
<td>2. Rendering the Five-Channel Volume</td>
<td>14</td>
</tr>
<tr>
<td>2.1. The Problem Introduced by this Requirement</td>
<td>14</td>
</tr>
<tr>
<td>2.2. Previous Work on the Visualization of Segmented Volumes</td>
<td>15</td>
</tr>
<tr>
<td>2.3. The Approach to Rendering the Five-Channel Volume</td>
<td>16</td>
</tr>
<tr>
<td>2.4. The Solution</td>
<td>17</td>
</tr>
<tr>
<td>2.5. Analysis of the Solution</td>
<td>24</td>
</tr>
<tr>
<td>3. Increasing Interactive Rendering Rates During Drilling</td>
<td>28</td>
</tr>
<tr>
<td>3.1. The Problem Introduced by this Requirement</td>
<td>28</td>
</tr>
<tr>
<td>3.2. The Approach to Increasing Rendering Rates</td>
<td>29</td>
</tr>
<tr>
<td>3.3. The Solution</td>
<td>30</td>
</tr>
<tr>
<td>3.4. Performance Results of the Solution</td>
<td>32</td>
</tr>
<tr>
<td>3.5. Analysis of the Solution</td>
<td>34</td>
</tr>
</tbody>
</table>

vii
4. Volume Haptics Using a Ray-Cast Force Field ........................................ 35
   4.1. Introduction .................................................................................. 35
   4.2. Previous Work on Volume Haptics .............................................. 35
   4.3. The Problem Introduced by this Requirement ................................ 37
   4.4. The Solution .................................................................................. 38
   4.5. Results of the Solution .................................................................. 42
   4.6. Analysis of the Solution ................................................................. 43

5. Current System Status ....................................................................... 45
   5.1. Description of the Current Simulator ........................................... 45
   5.2. Limitations of the Current System and Future Work ..................... 47

6. Conclusion ......................................................................................... 49

List of References .................................................................................. 51
## LIST OF TABLES

<table>
<thead>
<tr>
<th>Table</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1</td>
<td>Texture download performance of the development platform, with and without a hardware color lookup table enabled. Enabling the color lookup table reduces performance by approximately 50%</td>
<td>26</td>
</tr>
<tr>
<td>4.1</td>
<td>How the maximum stable spring length changes as the number of springs is increased</td>
<td>43</td>
</tr>
</tbody>
</table>
# LIST OF FIGURES

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.1</td>
<td>Location of the temporal bone (left lateral view of skull)</td>
<td>2</td>
</tr>
<tr>
<td>1.2</td>
<td>A typical temporal bone dissection workstation</td>
<td>3</td>
</tr>
<tr>
<td>1.3</td>
<td>Typical diamond (top) and tungsten-carbide (bottom) drilling burs</td>
<td>10</td>
</tr>
<tr>
<td>2.1</td>
<td>Two views of the same temporal bone data set, the left without per-voxel opacity information and the right with an 8-bit per-voxel opacity computed from neighborhood information. Note the increase in detail obtained by using neighborhood information.</td>
<td>18</td>
</tr>
<tr>
<td>2.2</td>
<td>Construction of the encoded alpha channel containing the quantized per-voxel opacity index, and the segment index</td>
<td>19</td>
</tr>
<tr>
<td>2.3</td>
<td>Two views of the temporal bone data set, the left with a 3-bit mapped per-voxel opacity and the right with an 8-bit per-voxel opacity. Note the nearly identical detail in the 3-bit image despite a reduction in opacity resolution.</td>
<td>20</td>
</tr>
<tr>
<td>2.4</td>
<td>Simplified flow diagram for a Color Lookup Table during pixel transfers in OpenGL 1.2</td>
<td>22</td>
</tr>
<tr>
<td>2.5</td>
<td>Simplified flow diagram for the construction of a single element of the master opacity table from the segment opacity table and voxel opacity table.</td>
<td>24</td>
</tr>
<tr>
<td>2.6</td>
<td>Sequence of images in which the opacities of multiple segmented structures are reduced to reveal the cochlea and semicircular canals within the volume of temporal bone.</td>
<td>25</td>
</tr>
<tr>
<td>3.1</td>
<td>View of temporal bone during a dissection. Note the relatively small size of the drilling bur compared to the size of the image.</td>
<td>29</td>
</tr>
<tr>
<td>3.2</td>
<td>Definition of the new view frustum from the global viewport and frustum.</td>
<td>32</td>
</tr>
<tr>
<td>Figure</td>
<td>Description</td>
<td>Page</td>
</tr>
<tr>
<td>--------</td>
<td>-----------------------------------------------------------------------------</td>
<td>------</td>
</tr>
<tr>
<td>3.3</td>
<td>Performance of the solution as a function of projected area. Areas below around 3600 pixels will achieve the desired rate of interactivity</td>
<td>33</td>
</tr>
<tr>
<td>4.1</td>
<td>A set of virtual springs uniformly distributed around the spherical representation of the drilling bur</td>
<td>39</td>
</tr>
<tr>
<td>4.2</td>
<td>Two-dimensional example of casting a ray using an integer scan-conversion (i.e. a digital differential analyzer) algorithm</td>
<td>41</td>
</tr>
<tr>
<td>4.3</td>
<td>$P_0$ and $P_1$ are the endpoints of the virtual spring and $d$ is its normalized direction. The displacement is used in Hooke’s Law and is computed as the dot product of $d$ and $(\text{center} - P_1)$</td>
<td>41</td>
</tr>
<tr>
<td>5.1</td>
<td>The virtual dissection workstation. Compare with the physical dissection workstation in figure 1.2</td>
<td>46</td>
</tr>
<tr>
<td>5.2</td>
<td>Screenshot of the simulator including hide-away in-context menus</td>
<td>46</td>
</tr>
</tbody>
</table>
CHAPTER 1

INTRODUCTION

The Temporal Bone Dissection Simulator is an ongoing research project for the construction of a synthetic environment suitable for virtual dissection of human temporal bone and regional anatomy. Funded by a grant (1-R21 DC04515-01) from The National Institute on Deafness and other Communication Disorders (NIDCD), the primary goal of this project is to provide a safe, robust, and cost-effective virtual environment for education of the anatomy and surgical procedures associated with the temporal bone in order to augment existing clinical curricula.

1.1 Background and Existing Methods

The temporal bones are located on the side of the human skull, above and behind the jaw and surrounding the ear canal (see figure 1.1). This area includes the auditory anatomy, the facial nerve, and vestibular organs and is the subject of many important and delicate procedures, such as exploratory and corrective surgery, biopsies, tumor removal, and implants [11, 19]. The anatomy of this region is very complex and contains many small, intricate, and fragile structures, leading to complex associated surgeries and
Figure 1.1: Location of the temporal bone (left lateral view of skull).

making it one of the most difficult areas of the body to understand [14]. Surgeons typically require many years of study to become familiar with the three-dimensional anatomic relationships between structures and to gain proficiency in temporal bone procedures [19]. Conventional training is achieved by means of printed material and sessions in a dissection lab, however these methods are both time consuming and expensive [11]. Kuppersmith et al. [11] and Nelson [19] provide an excellent description of the essential temporal bone dissection lab, which consists of one or more stations each with a temporal bone holder, a binocular microscope, a suction-irrigation device, and a drill (see figure 1.2). Using this equipment, a resident or surgeon performs a dissection by removing bone with the drill and removing fluid and bone dust with a suction-irrigation device. Structures are systematically exposed or removed in accordance with a manual or under direct supervision.
Figure 1.2: A typical temporal bone dissection workstation.

One drawback of a physical temporal bone dissection is that it requires the use of a dedicated laboratory and personnel as well as an adequate number of bone samples harvested from cadavers or synthesized as plastic models [11, 24]. Both requirements can incur substantial cost. Availability of cadaveric bone is inconsistent and typically limited to adult samples without interesting or unique pathologies, making it difficult to learn rare or specialized procedures or to practice on infant or child anatomy. Cadaveric samples also carry the risk of infection, and because dissections are usually limited to one per bone sample, mistakes can be costly [11]. Synthetic models offer a safer and lower cost alternative to cadaveric samples, but do not yet fully duplicate the behavior or pathological diversity of real bone. In addition, while dissection is the primary means for
surgical training, the dissection lab fails to fully reproduce the detail and scope of an in-vivo surgical environment [11].

The primary goal of this virtual dissection simulation is to provide a customizable and more complete alternative to learning anatomy and surgical technique as opposed to printed media or physical dissections. Mason et al. [14] and Harada et al. [8] stress the potential of simulations to provide a more thorough and rapid means of assimilating anatomy and procedure. Kuppersmith et al. [11] emphasize the ability of a virtual environment to allow repeated practice of complicated and pathology-specific procedures, as well as providing an objective and standardized means for evaluation. A virtual environment is also indispensable for pre-surgical planning, allowing a surgeon to practice techniques prior to a surgery. Each group also alludes to the importance of a virtual temporal bone for laying the foundation for surgical simulation.

1.2 Previous Work on Temporal Bone Simulations

A fair amount of research has emerged in recent years for visualizing and simulating interaction with virtual temporal bones. One of the first such systems was developed by Harada et al. [8], and used three dimensional reconstruction of histological cross sections of the temporal bone in order to increase understanding of the anatomy. The virtual bone could be arbitrarily oriented, sliced, and removed but required extensive preprocessing and was monochromatic. Kuppersmith et al. [11] provide excellent delineation for the construction of a virtual temporal bone dissection simulator, and have developed a preliminary system. Their system renders geometric surfaces that have been
extracted from segmented Computed Tomography (CT) and Magnetic Resonance Imaging (MRI) scans, and uses haptic and aural feedback to increase realism and provide validation for specific tasks. Mason et al. [14] have developed a virtual temporal bone system for use as an integrated teaching tool. Histological slices were scanned, aligned, manually segmented, then converted to geometric surfaces. The model was subsequently displayed in a CAVE™ environment [5] in stereo, and allowed a user to arbitrarily orient and alter the visibility of structures in order to learn the functional relationships of the anatomy while visualizing the bone in three-dimensional clarity. John et al. [9] are currently developing a petrous bone (a region inside the temporal bone) surgical simulator beginning with a detailed task analysis and leading up to the implementation of a system for pre-operative planning, surgical simulation, education, and training. As of this publication, preliminary results are still forthcoming. The project presented in this document is most similar to that of Kuppersmith et al. [11], but differs by using direct volume rendering for visualization and by incorporating aspects of an intelligent tutor.

1.3 Approach to the Construction of this Simulator

1.3.1 Key Components to the Effective Simulator

This simulation educates and trains users through tasks based on the identification and exposure of key structures. It is unique in that the system emphasizes performance evaluation in a surgical context by requiring a user to perform tasks consistent with an educational syllabus of the temporal bone region. One part of the system is the dissection
simulator for learning surgical procedures. The primary goal of this component is to gain experience and proficiency working with the temporal bone by manipulating a virtual drill and suction-irrigation device to appropriately expose specific structures. This part of the system is modeled after a physical temporal bone dissection workstation, such as that illustrated in figure 1.2. Another part of the system incorporates aspects of an intelligent tutor to provide a means for learning anatomy through interactive identification. Tasks performed in this part include identification of selected anatomy, manipulating the virtual bone to find and select specific anatomic structures requested by the tutor, and querying the tutor to locate specific anatomy. The simulation is designed to maximize transfer from the performance evaluation to subsequent surgical practice. This enables the user to learn relevant anatomy in a surgical context while providing preparation for subsequent surgical procedures. The working hypothesis during development of this system is that the two primary components, the dissector and the intelligent tutor, will together offer a more comprehensive and intuitive means for a student or resident to learn temporal bone anatomy and surgical procedure than conventional training methods.

1.3.2 Requirements of the Simulator System

The requirements of this project have been established by the project research team in conjunction with the NIDCD Project Investigator, and are motivated by conventional temporal bone dissection protocol [19], the results and recommendations of similar projects [8, 9, 11, 14], and first-hand experience by the research team in the Department of Otolaryngology Dissection Lab at The Ohio State University. The
technical requirements of the system have been chosen to provide enough realism to make the system functional as a replacement or augmentation to conventional training, while remaining technically feasible and cost-effective with current technology. The requirements also take into account the long-term extensibility of the project to surgical training and simulation.

The system must provide stereo presentation in order to duplicate stereoscopy from a binocular microscope. Two haptic (force-feedback) input devices are required to simulate the bone drill and suction-irrigation tool. Force-feedback must be provided for each input device so a user can effectively interact with the bone and perform the dissection. The system must also synthesize audio cues to simulate the pitch and frequency changes of the drilling bur during its interaction with bone. Most importantly, the system must provide a high degree of interactivity during the virtual dissection to eliminate user induced oscillations and control errors [23]. The goal for this system is to provide sustained frame rates between 15 and 20 frames per second during direct interactions (i.e. drilling and palpation) with the virtual bone.

The intelligent tutor portion of the simulator requires the ability to dynamically modify the opacities of many independent structures in order to emphasize and distinguish anatomy for the purposes of identification and comprehension. For example, the tutor will select and emphasize a specific structure by making it opaque and all other structures translucent. The tutor will then ask the user to identify the structure. Alternatively, the user can ask the tutor to locate a specific structure, at which time the tutor will emphasize it in the same manner.
1.3.3 System Hardware

The system was initially developed on a Silicon Graphics Onyx2™ with Infinite Reality2® graphics, four MIPS R10000 295 MHz processors, 1 GB main memory, and 64 MB texture memory. After initial development, the system was ported to a lower-cost Silicon Graphics Octane2™ system. Two PHANTOM™ haptic devices from Sensable Technologies [15] are being used for the force reflecting drill and the suction-irrigation tool. A Virtual Research V8 binocular display system is used to simulate the binocular microscope. For optimal compatibility with this display device, two 640x480 viewports are rendered by the system, one for each eye.

1.3.4 Data Used by the Simulator

The primary temporal bone data used by the simulator is an 80MB anisotropic volume comprised of 128x256x512 samples with a voxel resolution of 0.2x0.2x0.5 mm derived from CT and MRI acquisitions. Higher resolution volumes, including micro-CT acquisitions, will be obtained and incorporated later in the project. The acquired volume contains a single 12-bit channel representing the bone density at each sample point, and is quantized into one 8-bit channel to be more compatible with the visualization algorithm. Quantization of the data results in a net loss of information, however the capabilities of the system are not compromised. The volume is then segmented manually into various features by a team of experts to produce a single 8-bit channel volume of segment indices. Although auto-segmentation methods exist [22], they are not capable of functional segmentation. That is, automatic segmentation fails because there is no inherent
information in the volume data to delineate functional differences. For example, two adjacent structures could look the same but have distinct surgical significance, making them difficult or impossible to segment by a non-expert. For this reason, custom software was developed which allows experts to directly segment volumetric structures in stereo using the same visualization software used by the simulation. Each voxel in the segmented volume contains the segment index of the corresponding voxel in the bone volume. The acquired volume is then colored manually using filters, color tables, and direct “painting” of individual voxels to produce a standard three-channel RGB volume that more closely resembles actual physical anatomy. An opacity channel storing neighborhood information is then added to the color volume. This channel is used by the visualization algorithm to increase image quality, and is obtained by applying a non-linear filter to the occupancy information of the bone volume. The final temporal bone volume used by the simulator consists of five channels created from the original single channel: three color channels, an opacity (neighborhood) channel, and a segment index channel.

The drilling bur used in a real dissection is a small, spherical tungsten-carbide or diamond bit attached to a drill shaft (see figure 1.3). Typically, burs range in diameter from 1.0 mm to 7.0 mm in increments of 1.0 or 1.5 mm. Large tungsten-carbide burs are generally used for coarse and rapid bone removal, with smaller diamond burs used for more detailed work. In the simulator, the drilling bur is represented virtually by a sphere of appropriate size. The volume used in this project has a minimum resolution of 0.2 mm per voxel, which implies a 1.0 mm bur is about 5 voxels in diameter, and a 7.0 mm bur is
Figure 1.3: Typical diamond (top) and tungsten-carbide (bottom) drilling burs.

about 35 voxels in diameter. A cubic “modification kernel” encodes the virtual bur by using a spherical mask. Drilling is simulated by interactively positioning the modification kernel in the volume, with the mask controlling how voxels in the volume intersecting the kernel are modified. The kernel contains one of two values: the first value is assigned to the kernel region outside the mask and indicates no change; the second value is assigned to the kernel region inside the mask and indicates bone voxels should be removed (i.e. drilled away). Currently, removed voxels are simply made more transparent (i.e. their opacity decreased). In the future, removed voxels will be actively “thrown” based on the speed and orientation of the drilling bur, and propagated until they hit other stationary voxels, in order to simulate bone dust. The rate of voxel removal is a function of the
speed of the virtual drill, the size and composition of the bur, and the bone density of each voxel. Expert surgeons from the department of Otolaryngology at The Ohio State University have provided formative feedback for fine-tuning removal rates to increase accuracy.

1.3.5 Visualization Methods

Visualization is implemented using direct volume rendering methods [3, 4, 6, 10, 12, 20, 21, 27], rather than indirect methods [13], in order to reduce the amount of preprocessing overhead associated with local data changes and provide full visualization of all internal structures. Specifically, a three-dimensional texture-map based volume rendering algorithm [3, 4, 21] implemented in OpenGL 1.2 [28] is used to render the temporal bone data. This method was selected for several reasons: it is easy to implement; it makes direct use of acquired data with little to no preprocessing; it supports the frame rates necessary for effective interaction; no additional processing is required as a result of direct local or global changes to the data; and it is extremely efficient on the selected development platform. Other volume rendering methods [12, 13] require moderate to extensive preprocessing, which complicates algorithms for local modification and inhibits development, or are otherwise too slow under current implementations [5, 10, 27]. Meißner et al. [17] provide a thorough and accurate qualitative comparison of popular visualization techniques, including texture mapping, and is one source of justification for this choice. Direct rendering using dedicated volume rendering graphics hardware [20] is a promising technology that emerged after the start of the project, but is currently limited
to orthographic projection of 8 or 12 bit monochromatic data and consequently does not meet the requirements of this project. Future versions of this technology will be considered for possible porting to low cost platforms at the conclusion of the initial study.

1.4 Thesis Statement

This document gives an overview of the Temporal Bone Dissection Simulator, including its current status, and presents the primary problems and resulting solutions for satisfying three specific technical requirements of the system. At the start of the project, no known techniques were robust enough or produced adequate solutions or desired results for these problems. The first problem addresses the main visualization requirement of the intelligent tutor. The system requires the tutor to be capable of visibly differentiating multiple structures within the temporal bone region for the purposes of identification and assessment. This is achieved by dynamically adjusting the transparency of each structure. The problem this requirement introduces is how to make global transparency changes to our five-channel volume while remaining visually interactive. The technique in chapter 2 presents one way of accomplishing this goal. The second problem concerns the interactivity constraint of the simulator. The texture-map based visualization method chosen for the system, although fast, cannot maintain the required level of interactivity during drilling. The technique in chapter 3 describes how to augment the visualization algorithm to increase performance during drilling. The third problem involves the haptic requirements of the system. The simulator requires haptics to increase the overall realism and utility of the system, and to enable palpation of the virtual bone.
The technique in chapter 4 describes a volume haptic algorithm suitable for computing the forces required by the simulator. The overarching goal of the simulator is to create a performance evaluator that constrains the user to perform identification and exposure tasks within a surgical context mimicking that found in a physical temporal bone dissection. My thesis is that the virtual dissection simulation system will enable otolaryngological residents to increase their retention and rate of assimilation of temporal bone anatomy and surgical procedure, as compared to existing methods. Although some preliminary results have been obtained, this hypothesis will be thoroughly tested in an efficacy study performed at the Department of Otolaryngology at The Ohio State University concurrently with the continued development of the system.
CHAPTER 2

RENDERING THE FIVE-CHANNEL VOLUME

2.1 The Problem Introduced by this Requirement

One of the most important requirements of the intelligent tutor is the ability to quickly discriminate multiple structures within the volume of temporal bone. This feature is used to distinguish specific anatomy in order to test and train the user. For example, the assessment portion of the intelligent tutor will select and emphasize a structure, such as the Mastoid Process (an exterior portion of the temporal bone), and ask the user to identify it. The user is free to orient and slice through the Mastoid Process and examine it in the context of surrounding structures in order to identify it, and then enter an answer via keyboard. For this task, it is important that surrounding structures remain partially visible to preserve the anatomical context of the selected structure, and yet be partially transparent so as not to obscure it. To accomplish this, the tutor alters the opacity of each structure (that is, its visibility), making the selected structure fully opaque and all others partially transparent.

The development platform is well suited for interactively rendering up to four 8-bit channels of volume data of the size specified by the system requirements. However,
our volume containing the segmented anatomy of the temporal bone consists of five 8-bit channels. The key problem solved by the following technique is how to efficiently render the five-channel volume while at the same time providing the ability to interactively alter the transparency of multiple structures.

2.2 Previous Work on the Visualization of Segmented Volumes

A great deal of work has been done in the area of efficient and effective visualization of segmented or classified volumes. Some direct volume rendering techniques use ray-casting methods to visualize full-color segmented medical datasets. Tiede et al. [25] use volumes in which voxels exhibit binary membership (i.e. mutual exclusion) of each structure, with sub-voxel interpolation of a neighborhood determining final pixel color and opacity. Another technique [6], uses volumes with probabilistic or threshold classifications and successively composites view aligned volume slices to reconstruct the final image. In this technique, voxels can belong to more than one segment and rendered pixels receive a color and opacity that is a combination of the properties of each segment. While the images produced by these techniques are of high quality, performance is not in a real time or interactive range. Indirect volume rendering techniques [13] can be used to visualize segmented structures, particularly the boundaries between regions, at interactive rates. However they do not allow simultaneous visualization of the interior of structures. Several direct volume rendering techniques can also provide interactive and real time visualization of classified volumes. Hardware lookup tables implementing transfer functions and performing per-pixel classification are
popular in texture-map based volume rendering [18, 26]. Direct rendering with dedicated hardware [20] is another approach offering a great deal of flexibility and speed. The technique presented here is most similar to that of Meißner et al. [18] in that it relies on a hardware look-up table, but it is unique because it performs nonlinear opacity mapping dependent on two variables: the segment and the neighborhood. If only segment information was available, existing techniques, such as Meißner's, could satisfy the requirement of the tutor. However since per-voxel opacity information is needed to increase image quality, in addition to segment opacity information, no other technique provides the desired results.

2.3 The Approach to Rendering the Five-Channel Volume

The temporal bone data used in this project is acquired from CT and MRI scans. The volume consists of five channels, with three channels for color information, one for opacity (neighborhood) information, and one for segment information. Because the development platform is capable of rendering at most four channels of the volume with the desired performance, a method to encode all five channels of information into four channels is required for this application.

The final opacity for every voxel is dependent on both the per-voxel opacity derived from neighborhood information (stored in the opacity channel), and the opacity of the segment to which that voxel belongs (derived from the segment index channel). Specifically, the final opacity for each voxel is defined as the product of its original opacity, and the opacity of the segment to which it belongs. Due to the fact that both
channels affect the final opacity of each voxel in the bone volume, the devised solution combines the opacity channel and segment index channel into a single channel. This strategy is also directly compatible with the visualization hardware utilized by the system, and offers very little overhead. However, because only eight bits of space is available to store what originally required sixteen bits, some loss of information is necessary. This loss of information does not affect system capabilities due to the specific role of the opacity channel in the enhancement of image quality.

2.4 The Solution

The opacity channel is used to increase the quality of images produced by the visualization algorithm. To illustrate, figure 2.1 shows two views of the temporal bone dataset rendered by the system, the left rendered without opacity information and the right rendered with an 8-bit opacity computed from neighborhood information. The view without neighborhood information exhibits a great deal of aliasing and shows less detail than the view with neighborhood information. This reduction in quality occurs because of the way the texture-map based visualization algorithm reconstructs the volume of temporal bone. Specifically, it uses opacity information to blend slices of volume data together to create the final image. By using per-voxel opacities, the visualization algorithm is able to produce a much higher quality image. This level of quality is critical for precisely identifying the structures required by temporal bone procedures.
Figure 2.1: Two views of the same temporal bone data set, the left without per-voxel opacity information and the right with an 8-bit per-voxel opacity computed from neighborhood information. Note the increase in detail obtained by using an opacity channel storing neighborhood information.

The specifications of this system indicate that no more than 30 individual segmented structures will require visual discrimination during any given interactive session. Rather, of all segmented structures, different groups will be needed at different times according to the requirements of specific tasks. Using 32 as an upper bound on the number of segments, the segment index of each voxel can be represented using five bits. As there are eight bits available to encode both the segment and opacity of each voxel, this implies three bits (that is, 8 unique values) are left to encode opacity information. The manner in which the visualization algorithm uses the opacity channel during rendering requires that one segment value out of the 32 available and one opacity value out of the eight available be reserved for completely transparent voxels. Specifically,
completely transparent voxels containing no useful information are given a segment value and opacity value of 0. Therefore, at most 31 segment values and seven opacity values are available for non-transparent voxels.

The 8-bit opacity of each voxel is a number between 0 and 255. To reduce the opacity from eight bits to three bits, the original opacity is quantized by dropping the bottom five bits and keeping the top three bits, resulting in an opacity index value between 0 and 7. Depending on which anatomical structures are needed for the current task, up to 31 are selected and each voxel in the volume is assigned a unique 5-bit index representing the structure to which it belongs. Voxels that are not members of the 31 chosen segments are either made transparent by assigning an index of zero, or combined with another segment and assigned that segment’s index. The resulting 3-bit opacity index and 5-bit segment index are stored together in the top and bottom portions, respectively, of a new channel, the “alpha channel”.

![Diagram](image)

Figure 2.2: Construction of the encoded alpha channel containing the quantized per-voxel opacity index, and the segment index.
The visualization algorithm now has only three bits of opacity information to improve the quality of the final rendered image. During rendering, the 3-bit opacity value is used to index into an opacity table and retrieve an 8-bit opacity, which is close to the pre-quantized value. The left image in figure 2.3 illustrates the same temporal bone data set from figure 2.1 rendered with three bits of opacity information mapped by an appropriate opacity table. The right image shows the original 8-bit opacity image for comparison. Although there is some loss of quality, far more detail is present in the final 3-bit image than the image in figure 2.1 without opacity information. Image quality can be further improved by using fewer bits for segment information and more for opacity information, however this reduces the number of available segments. Preliminary feedback from the development team indicates that the choice of five bits for segment

Figure 2.3: Two views of the temporal bone data set, the left with a 3-bit mapped per-voxel opacity and the right with an 8-bit per-voxel opacity. Note the nearly identical detail in the 3-bit image despite a reduction in opacity resolution.
information and three bits for opacity information is an acceptable compromise between
utility and quality for the purposes of the intelligent tutor.

In a preprocessing step, the opacity and segment channels are reduced to three and
five bits, respectively, in the manner outlined above, and combined with the color
channels to create a volume with four channels: red, green, blue, and alpha (RGBA). The
alpha channel contains the 3-bit opacity index and the 5-bit segment index for each voxel.
This volume is rendered using a slightly modified version of the visualization technique
presented in chapter 1.

In order to make changes to the opacity of each segment, while incorporating per
voxel opacity information, the visualization algorithm is modified to utilize the color
lookup table feature of the OpenGL 1.2 API [28]. A lookup table is an array of values
which is used to replace other values during calculations. In OpenGL 1.2, the color
lookup table performs a mapping of red, green, blue, and opacity values for each pixel
element in a texture during pixel transfers (see figure 2.4). Prior to rendering, the color
lookup table is constructed to map the segment index and opacity (neighborhood) index
encoded in the alpha channel to the correct final opacity for each voxel. The visualization
algorithm used by the system transfers the temporal bone volume to a hardware resource
called texture memory as a stream of texels (texture elements) prior to rendering. During
this transfer, the color lookup table is applied, and each voxel in the volume is assigned
an opacity that is the product of its original opacity, and the opacity of the segment to
which that voxel belongs. The volume in texture memory is then rendered with the
texture-map based volume rendering algorithm used by the system.
Figure 2.4: Simplified flow diagram for a Color Lookup Table during pixel transfers in OpenGL 1.2.

To construct the color lookup table that performs the necessary mapping, two additional lookup tables must first be created, one for the 5-bit segment index and one for the 3-bit opacity index. The goal is to produce a single lookup table which contains the product of the segment opacity and per-voxel opacity for every voxel. The segment opacity table contains 32 8-bit entries, one for every 5-bit segment index. The voxel opacity table contains eight 8-bit entries, one for every 3-bit opacity index. Each entry in the segment opacity table contains the opacity of its corresponding segment, and can range from fully opaque to fully transparent. Each entry in the voxel opacity table contains an opacity used to increase rendering quality. Values in the segment opacity table are interactively modified by the system during a simulation to emphasize specific structures. For example, to emphasize the Mastoid Process, the system would increase its
visibility by increasing its opacity value in the segment opacity table, while simultaneously decreasing the opacity values of all other structures. Values in the voxel opacity table are subjectively chosen by hand in a preprocessing step to yield the best final image for the temporal bone volume. For a given volume, the voxel opacity table remains unchanged throughout the simulation.

Together, the segment opacity table and voxel opacity table are used to construct the *master opacity table* that performs the desired mapping. The master opacity table contains 256 8-bit entries, one for each possible value in the alpha channel. To construct the master opacity table, each 8-bit value in the alpha channel is first split into a 5-bit segment index and a 3-bit opacity index. The segment index is used to retrieve the opacity of the corresponding segment from the segment opacity table. The opacity index is used to retrieve a per-voxel opacity from the voxel opacity table. These two opacity values are multiplied together and stored in the master table at the location indexed by the 8-bit alpha value (see figure 2.5).

After the master opacity table is constructed, it is downloaded to OpenGL via the OpenGL function *glColorTable* [28], and the temporal bone volume is downloaded to texture memory. Prior to this download, the color lookup table is enabled (via *glEnable(GL_COLOR_TABLE)* ), allowing OpenGL to use the master table to map the incoming encoded alpha channel (containing segment and opacity indices) to the correct final opacity. These steps are executed only when changes have been made to the segment opacity table.
Figure 2.5: Simplified flow diagram for the construction of a single element of the master opacity table from the segment opacity table and voxel opacity table.

Figure 2.6 shows several images produced from this technique. The temporal bone volume displayed in these images contains 8 segmented structures including the mastoid process, semicircular canals, and cochlea. The sequence of images begins with all structures opaque, and successively shows several structures, including the mastoid process, rendered more transparent to reveal the cochlea and semicircular canals.

2.5 Analysis of the Solution

The results from this technique satisfy the primary visualization requirement of the tutor: the opacity of multiple segments can be modified interactively while incorporating per-voxel opacities. However, using this technique comes at a price: the
Figure 2.6: Sequence of images in which the opacities of multiple segmented structures are reduced to reveal the cochlea and semicircular canals within the volume of temporal bone.
volume needs to be downloaded for each change to the segment opacity table. If this
technique is not used, the entire temporal bone volume is downloaded only once at the
start of the simulation. Table 2.1 indicates how the use of this technique will impact
performance of the system. The table contains interactivity performance in frames per
second for texture downloads of multiple volume sizes, both with and without a color
lookup table enabled. The table contains download timings only and does not take into
account table construction time, table download time, or volume rendering time. This is
because, in all cases, table construction times and table download times are negligible
compared to the cost of downloading the volume. Rendering time varies with the current
viewing transformation, but results from the system indicate an average rendering rate of
2-3 frames per second.

<table>
<thead>
<tr>
<th>Volume Size</th>
<th>1MB</th>
<th>2MB</th>
<th>4MB</th>
<th>8MB</th>
<th>16MB</th>
<th>32MB</th>
<th>64MB</th>
</tr>
</thead>
<tbody>
<tr>
<td>Download Interactivity - Frames per second (without Color Lookup Table)</td>
<td>103</td>
<td>51</td>
<td>36</td>
<td>17</td>
<td>8.5</td>
<td>4.2</td>
<td>2.1</td>
</tr>
<tr>
<td>Download Interactivity - Frames per second (with Color Lookup Table)</td>
<td>65</td>
<td>33</td>
<td>17.6</td>
<td>8.63</td>
<td>4.11</td>
<td>2.09</td>
<td><strong>1.07</strong></td>
</tr>
</tbody>
</table>

Table 2.1: Texture download performance of the development platform, with and without a hardware color lookup table enabled. Enabling the color lookup table reduces performance by approximately 50%.
Timings from the table indicate that for the volume used by the system (64MB), overall frame rates during changes to segment opacities will be no more than one or two frames per second. Although this is much slower than the desired degree of interactivity, the specific use of this feature does not interfere with the utility of the tutor. Segment opacity changes are only made during transitions between identification problems and are presented as a brief animation. Although the animation itself proceeds at a slow rate, after the animation the volume does not need to be downloaded again and interactivity returns to that intrinsic to the visualization algorithm. After each problem is presented and the opacity changes are made, the user is free to interact with the virtual bone at this “normal” rate.
CHAPTER 3

INCREASING INTERACTIVE RENDERING RATES DURING DRILLING

3.1 The Problem Introduced by this Requirement

The most important requirement of the dissection simulator is the ability to allow the user to directly modify the virtual temporal bone. During a virtual dissection the user directs a modification kernel containing a spherical mask into the volume of temporal bone. The spherical mask represents a virtual drilling bur, and has a diameter based on both the desired virtual bur size and the voxel resolution. The resolution of the current volume and the useful range of burs dictate kernel sizes of at most $35^3$ voxels, with mask diameters ranging from 5 to 35 voxels. Hardware restrictions limit kernel sizes to be powers of two, so a $32^3$ kernel is actually used. As a consequence, the maximum virtual bur size is reduced from 7.0mm to 6.0mm. To simulate drilling, the mask is used to reduce the opacity of individual voxels, creating the illusion that they are being removed.

The dissection simulator is required to maintain stereo frame rates as high as possible, preferably at least 20 frames per second (in stereo) during drilling. The visualization method is currently not capable of achieving more than two or three frames per second in stereo due to rendering performance limitations in the visualization
hardware. In order for the simulation to be functional, a technique to increase the frame rate during direct interaction with the temporal bone, that is, during drilling and palpation, was required.

3.2 The Approach to Increasing Rendering Rates

Observations from a real temporal bone dissection reveal one way in which performance can be increased. In a dissection, the resident views the cadaveric sample through a binocular microscope and uses the drilling tool to remove small quantities of bone (see figure 3.1). During drilling, much of the image remains the same, with only the small region surrounding the bur changing visibly. In graphics terms, the viewpoint

![Image](image.jpg)

**Figure 3.1:** View of temporal bone during a dissection. Note the relatively small size of the drilling bur compared to the size of the image.
remains fixed while only small portions of the data are altered. This means the entire image does not need to be regenerated for every change to the bone, only those areas of the image that are directly affected by the drilling.

3.3 The Solution

The solution exploits the fact that the only portion of the image that changes is that contained in the two-dimensional projection of the modification kernel to the image. The following steps are repeatedly performed while drilling with the virtual bur:

1. Transform the virtual bur (i.e. the modification kernel) from viewing space to the object space of the volume
2. Update those voxels in the volume intersecting the kernel mask
3. Project the three-dimensional bounding box of the kernel to the screen
4. Calculate the 2D extents of the projected bounding box in screen space
5. Use the screen extents to calculate a new viewport and view frustum
6. Copy the previous frame’s rendering of the volume into the rendering buffer from a temporary buffer
7. Render the volume into the new viewport
8. Update the temporary buffer
9. Render the rest of the scene

In step 1 the transformations of both the PHANTOM™ input device and temporal bone volume are used to map the modification kernel into the object space of the volume. In step 2, a copy of the volume (which is stored in main memory) is updated based on the contents of the modification kernel, then downloaded to texture memory via the OpenGL call *glTexSubImage3D* [28]. This call transfers only the modified voxels to texture memory, thus avoiding the expense of transferring the whole volume. Step 3 requires projecting each vertex of the bounding box of the kernel into screen space. This is achieved by transforming the three-dimensional coordinates of the kernel by the world
viewing transformation and projection transformation, followed by a perspective divide and mapping to screen coordinates. For step 4, let \( x_0 \) and \( x_1 \) be the minimum and maximum x-coordinates of all projected vertices, and \( y_0 \) and \( y_1 \) be the minimum and maximum y-coordinates. The extent in screen space of the kernel is the rectangular region with lower left coordinate \((x_0, y_0)\) and upper right coordinate \((x_1, y_1)\). For step 5, assume the global viewport is defined by \((x, y, width, and height)\) as obtained from \( glGetInteger\) \((GL\_VIEWPORT, ...)\) \([28]\), and the global view frustum is defined by six numbers in the manner specified by \( glFrustum \) \([28]\): \((left, right, bottom, top, near, and far)\). The new viewport is simply the kernel extent in screen space, that is \((x0, y0, x1, y1)\), and the new view frustum is a linear interpolation from the global viewport and global view frustum (see Figure 3.2):

\[
\begin{align*}
\text{newLeft} &= \text{left} + (\text{right} - \text{left}) \times x_0 / \text{width} \\
\text{newBottom} &= \text{bottom} + (\text{top} - \text{bottom}) \times y_0 / \text{height} \\
\text{newRight} &= \text{newLeft} + (\text{right} - \text{left}) \times (x_1 - x_0) / \text{width} \\
\text{newTop} &= \text{newBottom} + (\text{top} - \text{bottom}) \times (y_1 - y_0) / \text{height}
\end{align*}
\]

In step 6, the image of the volume from the previous frame is copied into the rendering buffer from a temporary buffer. This is necessary because other information in the scene (e.g. the drill, the suction-irrigation device, text) obscures portions of the volume in the final image. Since the temporary buffer holds only the image of the volume, copying it produces the same results as rendering the whole volume, but without the cost of rendering the whole volume. In step 7, the new viewport is set via \( glViewport(x0, y0, x1, y1) \) \([28]\), and the new view frustum set via \( glFrustum(\text{newLeft}, \text{newRight}, \text{newBottom}, \text{newTop}, \text{near}, \text{far}) \) \([28]\). The volume is then rendered. The combination of the new viewport and new view frustum restrict rendering to only that portion of the image.
Figure 3.2: Definition of the new view frustum from the global viewport and frustum.

affected by changes due to the modification kernel. In step 8, the temporary buffer is updated with the contents of the new viewport so that it contains the correct final image of the volume. Step 9 then resets the viewport and view frustum and renders the rest of the scene.

3.4 Performance Results of the Solution

The solution was implemented on the development platform and its performance analyzed. Steps 1 through 5 of the algorithm had a constant cost of just over one millisecond for the $32^3$ subset (i.e. the modification kernel) of the 64MB RGBA temporal bone volume, which is negligible compared to the cost of steps 6 through 8. Figure 3.3 illustrates how the performance of steps 1 through 8 of the above algorithm changes as a
Figure 3.3: Performance of the solution as a function of projected area. Areas below around 3600 pixels will achieve the desired rate of interactivity.
function of the projected area of the modification kernel. Data in this figure was generated from the temporal bone volume covering approximately 80% of a stereoscopic image and rendered with a texture-map based algorithm and using 150 view-aligned slices in OpenGL. By noting where performance intersects the desired rate of interactivity (i.e. 20 frames per second), it is evident that the projected area of the modification kernel should be kept below approximately 3600 pixels (e.g. a 60x60 pixel rectangle) if the interactivity constraint is to be met.

3.5 Analysis of the Solution

Although this solution is capable of achieving the desired level of interactivity, performance is highly dependent on the projected area of the modification kernel and the current viewing parameters. To help ensure that performance remains at desirable levels, the system attempts to both restrict and adapt viewing parameters (including field of view, zoom, number of slices, etc.) based on the projected area of the modification kernel. In addition, because the spherical mask is the only part of the kernel that modifies the volume, the system uses the three-dimensional bounding box enclosing the spherical mask in step 3 of the solution, instead of the bounding box enclosing the whole kernel. In essence, smaller burs utilize a smaller modification kernel, which has the net effect of reducing the projected area of the modification kernel and improves performance.
CHAPTER 4

VOLUME HAPTICS USING A RAY-CAST FORCE FIELD

4.1 Introduction

In a dissection of the temporal bone, surgeons and residents use tactile feedback from the drilling bur to judge how much pressure to apply during removal and to palpate exposed anatomy. Haptic feedback, also known as force feedback or tactile feedback, is very important in this type of procedure because exposed structures are so small and delicate. Without the sense of touch, it is much more difficult to control the location and rate of bone removal, which can lead to dissection errors. Haptics is required by the dissection simulator to accurately reproduce sensations in a real dissection environment, and by the intelligent tutor to provide an extra modality to validate selection and reinforce visual learning.

4.2 Previous Work on Volume Haptics

Force feedback has been used for many years in entertainment, military, and industrial simulations. However, only recently has the advent of cheaper and more stable
commercial haptic systems enabled widespread incorporation of this form of feedback in medical and surgical simulations. Volume haptics, where a user is able to feel the topology and surface properties of volumetric data, is particularly difficult due to the vast amount of data in a typical volume and the update rates required by commercial haptic hardware. While volume haptics is still a fledgling field, several areas of research have produced promising results. Sensable technologies, producers of the PHANTOM™ [15] hardware used by this system, also produce the GHOST® software library for incorporating haptics into applications. With their software development kit, different force effects can be dynamically produced directly from the geometry in a scene.

Forces from volumes can be computed either directly or indirectly. Analogous to direct and indirect volume rendering, direct volume haptics computes forces directly from the voxel representation, while indirect haptics uses an intermediate representation. McNeely at al. [16] use a prototype six degree-of-freedom PHANTOM™ to compute direct volume haptics on arbitrarily shaped tools interacting with simulated aircraft components. Using a static voxel map of the volumetric scene and a point shell representation of the probe object, forces are computed by intersecting points in the shell with the voxel map, and using a six degree-of-freedom spring-damper system to perform the haptic calculations. Avila et al. [1] also use a PHANTOM™, but compute direct volume haptics by using a point contact model with volume density and gradient information obtained using transfer functions and preprocessing. Their goal is not to produce a realistic simulation of haptic interaction, but rather to convey information about the data not available from visualization alone. Blezek et al. [2] compute haptics from
medical images using indirect methods and a PHANTOM™ for force-reflection. Their strategy is to compute an implicit quadric surface from the voxels, and use a point contact model and spring calculations to compute the magnitude and direction of the applied force from the implicit surface. The solution presented below is most similar to that of McNeely et al. [16], but differs by accommodating dynamic changes to the volume and by using ray casting methods to compute intersections.

4.3 The Problem Introduced by this Requirement

Experience in a real temporal bone dissection lab indicates that the forces felt during drilling and palpation are largely directional (translational), that is, not rotational (torque). Therefore, forces in the system are modeled with only three degrees of freedom (3DOF). Fortunately, the PHANTOM™ hardware used by the system only supports three degrees of freedom of force feedback, so this constraint is easily accommodated by the system configuration.

The system requires directional forces to be computed on the virtual bur directly from the volume. A sphere of appropriate size is used to model the virtual drilling bur. The bur size can change during the simulation, but will be restricted to between 5 and 32 voxels in diameter based on the target volume resolution. Voxels in proximity to the virtual bur are modified frequently, and the haptic algorithm must be robust enough to allow for these dynamic changes. Additionally, the PHANTOM™ hardware requires all force calculations to complete in less than one millisecond.
One processor out of the four available on the development platform is reserved exclusively for the haptic algorithm. This choice was made to facilitate meeting the 1kHz update rate constraint imposed by the PHANTOM™ hardware. For the purposes of the haptic algorithm, surfaces in the volume are defined by the set of voxels with opacity greater than a system defined threshold.

4.4 The Solution

The haptic requirement of the system is satisfied by modeling a “force field” of virtual springs. The force field surrounds the virtual bur, extending outward from its surface. The field is given a size and stiffness that is dependent on the size and composition of the virtual bur. Each bur modeled by the system has a set of pre-selected values used to describe the force field. The virtual springs are used to maintain a minimum separation between the surface of the virtual bur and the surface of the volume.

The physical characteristics and behavior of springs are frequently idealized by Hooke’s Law, which can be found in virtually any physics textbook. Hooke’s Law is a linear equation approximating the force generated by compressing or stretching a spring:

\[ F = -k \Delta x \quad \Delta x = d(\Delta l) \quad \Delta l = l_0 - l, \]

\( l_r = \text{spring rest length}, \quad l_0 = \text{current length of spring}, \)

\( \Delta l = \text{change in length}, \quad d = \text{normalized spring direction} \)

In its simplest form, Hooke’s Law is dependent on two quantities: the stiffness (or spring) constant \( k \), and the displacement \( \Delta x \). The equation states that the force \( F \) resulting from compressing or stretching a spring is proportional to the displacement from its rest
length ($\Delta l$). In vector form, the final force applied by a spring has magnitude equal to the
displacement of the spring times the spring constant, and is exerted along the direction of
the spring ($\mathbf{d}$).

In a preprocessing step, a set of virtual springs is distributed uniformly on the
surface of the sphere representing the virtual bur (see figure 4.1). One end of each virtual
spring is attached to the surface of the bur, and the other extended, normal to the surface,
out to the size of the force field. The unit normal of the sphere is computed for each
anchor point and assigned to the corresponding spring as its direction. The rest length of
each spring defines the size of the force field, and the spring constant of each spring
defines its stiffness.

Figure 4.1: A set of virtual springs uniformly distributed
around the spherical representation of the drilling bur.
After preprocessing, the haptic loop is spawned as a separate thread of execution from the main simulation process. The loop computes forces on the virtual bur due to the volume by executing the following steps for each haptic cycle:

1. Transform each spring into the object space of the volume
2. Cast one ray into the volume for each spring, and intersect it with the volume to find the distance to the nearest surface voxel
3. Compute forces for each spring based on its distance to the surface, and accumulate the total force
4. Apply the total force to the haptic hardware

In step 1, both the current volume transformation and the PHANTOM™ transformation are used to map each spring into the object space of the volume. The endpoints of each spring are calculated, and in step 2 a ray is cast out from the anchor endpoint in the direction of the other endpoint. Using an extended version of an integer scan conversion algorithm (i.e. a digital differential analyzer (DDA)) [7], each ray is efficiently extended through the volume until it intersects a surface or reaches the second endpoint (i.e. the size of the force field, see figure 4.2). When a surface voxel is reached, the distance from the virtual bur to the surface voxel is computed. For speed, this distance is approximated by the distance the anchor is to a plane through the center of the voxel and orthogonal to the ray (see figure 4.3). In step 3, this distance is used in an evaluation of Hooke’s Law to approximate the force exerted on the virtual bur by the virtual spring. The force for each spring is then accumulated to obtain the total force on the virtual bur, which is reflected by the PHANTOM™ in step 4.
Figure 4.2: Two-dimensional example of casting a ray using an integer scan-conversion (i.e. a digital differential analyzer) algorithm.

Figure 4.3: \( P_0 \) and \( P_1 \) are the endpoints of the virtual spring and \( d \) is its normalized direction. The displacement is used in Hooke's Law and is computed as the dot product of \( d \) and \( (\text{center} - P_1) \).
4.5 Results of the Solution

The solution was implemented on the development platform using a PHANTOM™ 1.5 and the GHOST® SDK, both provided from Sensable Technologies. The software library contained in the SDK maintains a strict 1 kHz haptic calculation loop synchronized with the haptic hardware. This means that each execution of the above algorithm must complete in less than one millisecond. The performance of the above algorithm is dependent on both the number and length of the virtual springs. The modification kernel is known to be no larger than 32 voxels in diameter. This implies that the longest spring length is no longer than the radius of the largest bur, or 16 voxels. In general, increasing the number of springs improves the haptic sensation by allowing more accurate force calculations. However, increasing the number of springs also reduces the maximum length of the springs because of the limited amount of computation time available. In order to determine the best combination of spring number and length for this project, that is, the maximum stable spring length for a given number of springs, four separate trials were conducted. In the first trial, 66 springs were distributed on a sphere, and the haptic algorithm executed. The GHOST® SDK reports an error if it detects that the haptic calculation loop is exceeding the one millisecond restriction. The rest length of the springs was repeatedly incremented until the error was reported, and then the current length recorded. This procedure was then repeated for 100, 150, and 200 springs. Table 4.1 summarizes the results of these trials.
<table>
<thead>
<tr>
<th>Number of Springs</th>
<th>66</th>
<th>100</th>
<th>150</th>
<th>200</th>
</tr>
</thead>
<tbody>
<tr>
<td>Maximum Stable Spring Length (voxels)</td>
<td>32</td>
<td>22</td>
<td>13</td>
<td>8</td>
</tr>
</tbody>
</table>

Table 4.1: How the maximum stable spring length changes as the number of springs is increased.

These trials resulted in the selection of a force field with 100 springs. The maximum stable spring length of 22 voxels for this number of springs exceeds the maximum length required by the system by a small margin. This margin was left to ensure that load balancing or operating system overhead would not inadvertently cause the haptic calculation loop to exceed the one millisecond restriction.

4.6 Analysis of the Solution

In general, the forces produced by this solution were judged to be good for simulating contact between a virtual sphere and a volumetric surface. This conclusion was reached through formative feedback from several otolaryngology residents and surgeons. The Project Investigator, Dr. Gregory Wiet, was instrumental in fine-tuning the force-field parameters for each virtual bur used by the system. Responses from several otolaryngology surgeons with no connection to the project have also been very positive.

The algorithm performs quite well for larger bur sizes (12 to 32 voxels in diameter), however, forces become increasingly unstable for smaller burs (less than 12 voxels in diameter). As the rest length of the virtual springs approaches five voxels, forces become more discontinuous and much less accurate and aesthetically pleasing.
This problem is partially attributed to the inaccuracy of the current DDA algorithm for detecting the nearest surface in all situations. Since the algorithm skips some voxels the true ray passes through, it will occasionally miss the nearest intersection altogether during a given cycle, but find it during the next. If a close intersection is skipped one cycle and found the next, it introduces a force discontinuity on the corresponding spring. To mitigate this problem, a new DDA algorithm will be constructed to check every voxel the true ray passes through. In addition, the spring constant and force field density will be adjusted dynamically to achieve the most stable haptic response for each bur size.
CHAPTER 5

CURRENT SYSTEM STATUS

5.1 Description of the Current Simulator

Although this project has one year remaining in a two-year grant, most of the significant technical challenges have been solved. The dissection portion of the system is functional and supports most of the required tasks. The system is capable of simulating an arbitrary dissection of virtual temporal bone, sufficient for performing a basic mastoidectomy and exposing key anatomy. The simulation provides stereo presentation on the binocular display device with both aural and haptic feedback. Figure 5.1 illustrates the virtual dissection workstation, which has duplicated the critical parts of the physical dissection workstation shown in figure 1.2. Although the current system has not yet fully integrated with the intelligent tutor portion, many aspects of the tutor have been incorporated. The user can navigate an in-context graphical menu interface to activate several aspects of the tutor, including the basic assessment and querying capabilities. Figure 5.2 shows a view of the simulator as it is presented to the user via the binocular display. The in-context menus are unobtrusively placed on the bottom and at the corners of the viewable scene and scroll out of sight when not in use.
Figure 5.1: The virtual dissection workstation. Compare with the physical dissection workstation in figure 1.2.

Figure 5.2: Screenshot of the simulator including hide-away in-context menus.
The current system has received strong preliminary support from residents and expert surgeons and construct validity has been obtained through formative evaluations with experts. Additional assessments obtained from an efficacy study with surgical residents throughout the second year of the project will provide information on criterion–related validity.

5.2 Limitations of the Current System and Future Work

Although achieving high, sustained frame rates remained a fundamental goal throughout development, the system just meets the minimum interactivity requirement. Interactive rates during bone removal peak at around 20 frames per second, with an average around 16 frames per second. During global transformations, frame rates drop to between 5 and 8 frames per second. Interactivity during global changes would normally be reduced to that intrinsic to the visualization algorithm (between two and three frames per second), however in an effort to increase interactivity during global transformations, creative use of texture mapping enables performance to be enhanced with only a slight loss in image quality.

The suction-irrigation device has not been integrated due to the lack of support for bone dust and fluid calculations. While these features are an integral part of a temporal bone surgery, and do exist to some extent in a dissection, for this project they have been classified as aspects of realism and have been assigned a low priority. Due to the ancillary nature of these features for identifying and exposing structures, they will be implemented only if development time allows after more pressing features are incorporated.
While the first year of the project focused primarily on solving fundamental and specific technical requirements, the second year will focus on integrating evaluation protocols, constructing a more functional user interface, and obtaining higher resolution and pathologically diverse data. The simulator will be enhanced to both record and play back user sessions in the simulator, and to compare dissection results with expert models for assessment. A more elaborate user interface will be constructed to allow easy navigation of the system from within the binocular display device. This will allow a user to change options and interact with the tutor without leaving a surgical context. Finally, the simulator will be integrated into the dissection curriculum of the Department of Otolaryngology at The Ohio State University, and an efficacy study performed with surgical residents to determine how well the simulator functions as an augmentation to conventional training.

Continued development and refinement of the system will allow the addition of many aspects of realism, including fluid dynamics from bleeding and the suction-irrigation device, bone discoloration due to overheating, bone paste and dust calculations, and voice recognition for easier interaction with the tutor portion. Muscle, arteries, and other soft tissue play a smaller role than bone in dissections and surgeries of the temporal bone region, however their existence cannot be ignored. Future research will explore the addition of real-time, dynamic models for soft tissue, and may achieve the ultimate goal: the simulation of in-vivo temporal bone surgery.
CHAPTER 6

CONCLUSION

The Temporal Bone Dissection Simulator project was introduced, including a brief explanation of similar projects and motivation for research of simulated surgical procedures. Three specific technical requirements of the system have been presented, along with their principal problems and current solutions. The existing simulator was described and its chief limitations identified.

One technique was presented for both rendering and interactively altering the opacity of multiple segments in a segmented, five-channel volume. This achievement allows aspects of an intelligent tutor to more thoroughly train and assess a user in the system. A second technique was described for increasing rendering performance during highly localized interaction with virtual anatomy. This accomplishment removed the most significant obstacle withholding the goal of our virtual dissection: that of sufficient interactivity to provide a convincing illusion for the user. A third and final technique was presented for computing force-feedback on a dynamically changing volume. This feature is essential for allowing the user to realistically interact with virtual anatomy.
In summary, a working prototype system for the virtual simulation of temporal bone dissection has been presented. The system offers a paradigm from traditional training practices by integrating advanced technology to provide a safer and more cost-effective way to learn fundamental techniques used in temporal bone surgeries. Through the use of a real-time, interactive volumetric simulation, the system obviates the need for physical materials in initial training and provides a more accessible way for residents to both practice technique and increase exposure to pathological variance of anatomy. Through continued refinements and evaluation, the system will help investigate the extensibility of virtual environments to preoperative assessment, presurgical planning, and surgical documentation.
LIST OF REFERENCES


51


