Virtual Cadaver Navigation System: Using Virtual Reality For Learning Human Anatomy

by

Abhijit V. Lothe

A thesis submitted in partial fulfillment
of the requirements for the degree of
Master of Science in Computer Science
Department of Computer Science and Engineering
College of Engineering
University of South Florida

Major Professor: Nagarajan Ranganathan Ph.D.
Don Hilbelink, Ph.D.
Sudeep Sarkar, Ph.D.

Date of Approval:
June 9, 2005

Keywords: Virtual Reality, Visible Human, 3D texture mapping, Gigabyte Volume Exploration, Direct Volume Rendering

DEDICATION

To my Parents and loving sister Gauri.

ACKNOWLEDGEMENTS

I would like to thank my major professor Dr. Ranganathan who kindly allowed me to pursue my research interest and helped me at every step as a good friend. I am very thankful to him for the numerous discussions and meetings we had towards designing an efficient VCNS system. My sincere thanks to Dr. Hilbelink for providing his access to his lab facilities, the hardware motion tracker system, and the guidance towards developing the system. I would also like to thank him for providing me an access furnishing the hardware and the software required to implement the system. I am grateful to Dr. Sarkar for taking time out of his busy schedule to review my thesis and provide useful comments to improve the same.

TABLE OF CONTENTS

LIST OF TABLES iii

LIST OF FIGURES iv

ABSTRACT vii

CHAPTER 1 INTRODUCTION 1
- 1.1 Definition of Virtual Reality 1
- 1.2 Key Elements of Virtual Reality Systems 2
  - 1.2.1 Virtual World or Environment 2
  - 1.2.2 Sense of Immersion 2
  - 1.2.3 Sensory Feedback 2
  - 1.2.4 Interaction 3
- 1.3 Applications of Virtual Reality 3
- 1.4 Virtual Reality in Medical Training and Education 3
- 1.5 Virtual Cadaver Navigation System 5
  - 1.5.1 Motivation for this Work 5
- 1.6 Contributions of Thesis 8
- 1.7 Organization of Thesis 8

CHAPTER 2 LITERATURE REVIEW 10
- 2.1 Surgical Planning and Procedures 10
- 2.2 Virtual Endoscopy 12
- 2.3 Medical Education 12
- 2.4 Neuropsychological Rehabilitation and Psychology 13
- 2.5 Issues and Challenges in using VR for Healthcare 14
- 2.6 Visualization of Large Volume Data 15
- 2.7 Context of this Work 16

CHAPTER 3 VISIBLE HUMAN DATASET 17
- 3.1 Visible Male 17
- 3.2 Visible Female 18
- 3.3 Segmented Dataset 18
- 3.4 Data Format 18

CHAPTER 4 ARCHITECTURE OF VCNS 21
- 4.1 Modes of Operation 21
- 4.2 Software Architecture 23
  - 4.2.1 Data Management Module 23

4.2.1.1 Data I/O 23
4.2.1.2 Resource Manager 24
4.2.2 Visualization Module 25
4.2.3 Tracking Module 25
4.2.4 Centroid Locator Module 26
4.2.5 Labeling Module 26
4.2.6 Collision Detection Module 27
4.3 Data and Computational Flow of VCNS 29
4.3.1 Tracking Mode 29
4.3.2 Centroid Locator Mode 32
4.3.3 Labeling Mode 32
4.4 Hardware Components of the System 35
4.4.1 PCI Bird Motion Tracker 35
4.4.1.1 Transmitter 35
4.4.1.2 Sensor 35
4.4.1.3 Electronics 36
4.4.1.4 Measurement Technique 36
4.5 Graphical User Interface 37

CHAPTER 5 SOFTWARE IMPLEMENTATION 40
5.1 Class Design 40
5.1.1 *PCIBirdInterface* Class 40
5.1.2 *MemoryManager* Class 42
5.1.3 *VolumeDataLoader* Class 44
5.1.4 *CollisionDetector* Class 44
5.1.5 *Renderer* Class 45
5.1.6 *LabelGenerator* Class 46
5.1.7 *Manager* Class 46
5.2 Software Implementation 47
5.2.1 Tracking Mode 47
5.2.2 Centroid Location Mode 49
5.2.3 Labeling Mode 49

CHAPTER 6 RESULTS AND DISCUSSION 52
6.1 Results 52
6.2 Performance Analysis 52
6.3 Discussion 59

REFERENCES 61

APPENDICES 66
Appendix A Class Diagram for VCNS 67

LIST OF TABLES

Table 1.1	Applications Of Virtual Reality	3
Table 4.1	Specifications Of Workstation	37
Table 6.1	Configurations Of Two Test Machines Used For Performance Analysis	56

LIST OF FIGURES

Figure 1.1	The Diagram Shows How VCNS Can Be Used To Obtain Cross Sectional View (Tomographic Slice) Of Virtual Human Body At Arbitrary Locations And Orientation During The Tracking Mode	7
Figure 3.1	Visible Human Male Images (a) MRI Image (b) CT Image (c) High Resolution Colored Image	19
Figure 3.2	Segmented Transaxial Slice Through Visible Male Thorax	19
Figure 3.3	A Transaxial Slice Through Visible Female Head	19
Figure 4.1	Software Architecture Of VCNS Showing Different Modules, And Inputs And Outputs Of The System	22
Figure 4.2	Software Architecture Of Data Management Module	24
Figure 4.3	Software Architecture Of Visualization Module	24
Figure 4.4	Software Architecture Of Tracking Module	25
Figure 4.5	Software Architecture Of Centroid Locator Module	26
Figure 4.6	Software Architecture Of Labeling Module	27
Figure 4.7	Binary Space Partition Tree	28
Figure 4.8	Collision Detection Process	28
Figure 4.9	Computational Flow Of VCNS	30
Figure 4.10	Computational Flow Chart And Data Flow Diagram For Tracking Mode Of VCNS	31
Figure 4.11	Computational Flow Chart And Data Flow Diagram For Centroid Mode Of VCNS	33
Figure 4.12	Computational Flow Chart And Data Flow Diagram For Labeling Mode of VCNS	34
Figure 4.13	Reference Frames For (a) Standard Transmitter (b) 25 mm sensor (Courtesy: Ascension Technology)	36

Figure 4.14 User Interface For VCNS 39

Figure 5.1 Interaction Diagram For Software Components Of VCNS 41

Figure 5.2 Conversion From PCIBird To OpenGL Coordinate System 42

Figure 5.3 Hierarchial Distribution Of Volume Data In VCNS 43

Figure 5.4 Sequence Diagram For Tracking Mode Based On UML Principles. An Underline Below A Class Name Implies That The Instance Of That Class Is Used 48

Figure 5.5 Sequence Diagram For Labeling Mode Based On UML Principles. An Underline Below A Class Name Implies That The Instance Of That Class Is Used 50

Figure 6.1 A Tomographic Slice Of Lung In Tracking Mode Through Visible Male Data 53

Figure 6.2 A Tomographic Slice Of Vertebral Column In Tracking Mode Through Visible Male Data 53

Figure 6.3 Identification Of Lower Left Lobe Of Lung Using The Labeling Tool 54

Figure 6.4 Diagram Showing The Intersection Of The Three Orthogonal Planes Through The Visible Male During Centroid Locator Mode 54

Figure 6.5 Transaxial Slice Through The Visible Male Data Generated During Centroid Locator Mode 55

Figure 6.6 Sagittal Slice Through The Visible Male Data Generated During Centroid Locator Mode 55

Figure 6.7 Coronal Slice Through The Visible Male Data Generated During Centroid Locator Mode 55

Figure 6.8 Variation Of Frame Rate With Texture Memory Size On NVIDIA Quadro FX 4400 (Main Memory = 512MB, Brick Size = 64 X 64 X 64) 56

Figure 6.9 Variation Of Frame Rate With Texture Memory Size On ATI Radeon 8700 (Main Memory = 512MB, Brick Size = 64 X 64 X 64) 57

Figure 6.10 Variation Of Frame Rate With Brick Size On NVIDIA Quadro FX 4400 (Main Memory = 512MB, Texture Memory = 64MB) 57

Figure 6.11 Variation Of Frame Rate With Brick Size On ATI Radeon 8700 (Main Memory = 512MB, Texture Memory = 64MB) 57

Figure 6.12 Variation of Frame Rate With Main Memory Size On NVIDIA Quadro FX 4400 (Texture Memory = 64MB, Brick Size = 64 X 64 X 64) 58

Figure 6.13 Variation Of Frame Rate With Main Memory Size On ATI Radeon 8700 (Texture Memory = 64MB, Brick Size = 64 X 64 X 64) 58

Figure A.1 VCNS Class Diagram 68

VIRTUAL CADAVER NAVIGATION SYSTEM: USING VIRTUAL REALITY FOR LEARNING HUMAN ANATOMY

Abhijit V. Lothe

ABSTRACT

The use of virtual reality (VR) for visualization can revolutionize medical training by simulating real world medical training procedures through intuitive and engaging user interface. Existing virtual reality based visualization systems for human anatomy are based on 3D surface and volumetric models and simulative systems based on model libraries. The visual impact as well as facilitation for learning are inadequate in such systems. This thesis research is aimed at eliminating such inadequacies by developing a non-immersive virtual reality system framework for storage, access and navigation of real human cadaveric data. Based on this framework, a real time software system called virtual cadaver navigation system (VCNS) is developed, that can be used as an aid for teaching human anatomy.

The hardware components of the system include, a mannequin, an examination probe similar to a medical ultrasound probe, and a personal computer. The examination probe is moved over the mannequin to obtain the virtual tomographic slice from the real cadaveric 3-D volume data. A 3-D binary space partitioning tree structure is defined to organize the entire volumetric data, by subdividing it into small blocks of predefined size, called as "bricks" that are assigned a unique address for identification. As the examination probe is moved over the mannequin, the set of bricks intersecting the corresponding tomographic slice are determined by traversing the tree structure, and only, the selected bricks are accessed from the main memory and brought into the texture memory on the graphics accelerator card for visualization. The texture memory in the graphics card and the main memory are divided into slots of size, that is a multiple of the brick size, and a tagging scheme that relates the brick addresses, texture memory slots, and the main memory blocks

is developed. Based on spatial, temporal and sequential locality of reference, only the currently required bricks as well as some of the neighboring bricks are loaded from the main memory into the texture memory, in order to maintain the highest frame rates required for real time visualization. The above framework consisting of the data organization and the access mechanism are critical in terms of achieving the interactive frame rates required for real-time visualization.

The input data to the system consists of non-segmented voxel data, and the data segmented and labelled based on tissue classification. The software system includes a labeling tool, in order to display the specific tissue information at the the location of the mouse cursor. This facility is useful in both teaching anatomy and self learning. Thus, the proposed VCNS system supports efficient navigation through the human body for learning anatomy and provides the knowledge of spatial locations and the interrelationship among the various organs of the body. A prototype software system has been developed, which is capable of achieving a throughput of 30 frames per second and has been tested with a 18-Gigabyte human cadaveric data obtained from the National Library of Medicine, on a personal computer with 64 Megabytes of texture memory and 512 Megabytes of main memory.

CHAPTER 1

INTRODUCTION

Human history is marked by the development and evolution of different communication media. From the age old cave paintings to the most recent virtual reality (VR), there has always been a quest for new and effective ways to convey the ideas. In the last decade, the field of VR has been explored because of its suitability for presenting new and existing information in a more intuitive way. The origin of the VR is believed to be in the flight simulator systems developed to train the pilots by putting them in a "real-like" environment. So what is VR after all? Since VR is a new medium, its definition is still in flux. Several definitions of the VR have been presented by the researchers and users from their own perspective. Some of them are as follows.

1.1 Definition of Virtual Reality

"An artificial environment which is experienced through sensory stimuli (as sights and sounds) provided by a computer and in which one's actions partially determine what happens in the environment" - Merriam-Webster [4].

"Virtual reality is a medium composed of interactive computer simulations, that sense the participant's position and actions and replace or augment the feedback to one or more senses, giving the feeling of being mentally immersed or present in the simulation (a virtual world)" - [53].

From these and other similar definitions four key elements in a VR system can be identified. They are as follows [53, 59].

1.2 Key Elements of Virtual Reality Systems

1.2.1 Virtual World or Environment

It is a manifestation of an imaginary world or a real space that exists elsewhere. In case of VR, it is made up of virtual objects (VO), which are governed by certain rules and relationships between them.

1.2.2 Sense of Immersion

The participant in the VR can either get completely (Immersive) or partially immersed (Non-Immersive) in the virtual world, depending on whether he is completely isolated from the real world. For example, the head mounted displays (HMD) provide the user with a personal view of the virtual environment, making the user unaware of the real world. On the other hand, non-immersive systems leaves the user visually aware of the real world, however, they are able to see the virtual world through some display device such as a graphics workstation [59]. The hybrid systems, also known as "Augmented Reality Systems", superimpose the real world view with the synthetic images obtained from the virtual world. Some VR systems allow only one user to be immersed in the virtual environment, while collaborative systems involve more than one participant interacting with the system, or with other participants in the virtual world.

1.2.3 Sensory Feedback

Unlike the traditional communication media, sensory feedback is an essential ingredient of VR systems. Although, the visual feedback has been most dominant, others include auditory [12], olfactory [30] and haptic (touch) [29].The feedback is generally based on the position of the participant, which necessitates the tracking of their movements. The tracking involves computerized sensing of the position (location and/or orientation) of the participant's body, or at least a part of his body.

1.2.4 Interaction

This involves the ability to affect the virtual world through the actions in the real world. For example, a VR system may allow picking up objects, setting them down or flipping switches etc. Interactivity gives authenticity to the VR systems.

1.3 Applications of Virtual Reality

There are numerous practical applications for which VR has been able to provide more interactivity and better interpretations of complex data than possible before. Table 1.1 lists the applications in different areas to which the VR has been applied successfully. The scope of VR, however, is endless.

Virtual reality is gaining recognition for its enormous educational potential particularly in the medical surgical simulations and related techniques. The following section gives an overview of the state of the art in the VR based medical training systems. A more detailed discussion is presented in Chapter 2.

Table 1.1. Applications Of Virtual Reality

Areas	Applications of VR
Engineering	Aero-engine design, Submarine Design, Virtual car prototyping, Architectural designs of buildings and rooms
Entertainment	Computer animation of cartoon characters, VR based games and theaters, Realtime cartoon animations
Science	Molecular modeling, Telepresence-controlled Remotely Operated Vehicle, Ultrasound echography, Visualization of electric fields
Training	Flight simulations, Fire fighter training, Virtual surgery Colonoscopy, Military training, Nuclear accident simulation Power plant simulation

1.4 Virtual Reality in Medical Training and Education

The VR has been used in the healthcare in three areas: surgical simulation and planning, medical education, and recently in the neuropsychological assessment and rehabilitation. The earliest use of VR in medicine dates back to early 90's and focused on the 3D

visualization of complex medical data for surgery and surgical planning [15]. These systems have been augmented by more sophisticated and interactive systems that integrate sensory feedback such as the haptic (touch) feedback, moving the user of the system closer to realism. Already, advanced simulation systems using various implementation strategies and targeting different areas of the human body have been developed [43, 49, 37]. For example, the Minimally Invasive Surgery Training-Virtual Reality (MIST-VR) trainer [10, 24] has been proved to be more effective in performing the laparoscopic skills than the traditional methods. Virtual endoscopy is another area where virtual reality has been used to solve problems faced with real endoscopy training procedures. A virtual endoscopy simulator allows the students to fly inside the organs by reconstructing the virtual surface models in real time. The surgical planning procedures, which typically involve study of series of 2D images of different modalities (CT/MRI), can also be enhanced by a VR system that integrates the modalities from different sites to provide an interactive three-dimensional view [46]. VR has also been helping the clinical psychologists by simulating the real world, which can be fully controlled by the user through various parameters. A key advantage offered by VR in this case, is to provide the patient an ability to successfully manage situations related to his/her disturbances [61, 20].

Complex topics such as human anatomy, biochemistry and molecular biology have been made more comprehensible with the intuitive and engaging VR based teaching environments. The first step in this direction was the creation of the VR anatomy books, which contained images from a real human being as a part of Visible Human project conducted by National Library of Medicine in 1993 [7, 55]. Through the 3D visualization of massive volumes of information and databases, clinicians and students can understand important physiological principles or basic anatomy [9]. Such VR systems can be used as educational tools allowing a deeper understanding of interrelationships between the anatomical structures that cannot be achieved by any other means, including cadaver dissection [43]. Apart from teaching anatomy, VR has been used for teaching 12-lead ECG [28, 58].

Virtual reality has, thus, broadened the overall training experience for the medical students by providing them an ability to acquire proficiency and confidence in performing

variety of techniques before they can do it clinically. The simulation systems in combination with the training on real patients can enhance the acquisition of clinical skills and increase the depth and the breadth of knowledge about the medical problems.

1.5 Virtual Cadaver Navigation System

1.5.1 Motivation for this Work

In a typical anatomy learning session involving cadaveric dissection, an instructor teaches the students a particular organ in isolation and then, in relation to the other parts in its vicinity or with respect to some landmark features surrounding it. During the evaluation, the students are expected to identify these structures by relating to the associations taught to them. Thus, understanding the inter-relationships between various organs and tissues is important in learning human anatomy.

Due to the dearth in number of cadavers available for dissection, sometimes the students are taught on the previously dissected cadavers that are devoid of the key landmarks. In addition, they find it difficult to locate some anatomical parts such as nerves, when presented with a live human body. Thus, there is a need to develop a method by which the locations of various organs, nerves and tissues inside the body as well as their relative positions can be understood in a way better than cadaveric dissections.

Previously, Teistler et. al. [57] has developed a system for learning the anatomy by simulating an ultrasound-like navigation of the 3D VOXELMAN dataset [27]. A virtual examination probe (analogy: medical ultrasound probe), is used to generate oblique tomographic images that are computed from a given volume data [57]. This system, however, does not identify the anatomical parts on the slice obtained from the tracker, which is imperative for learning/teaching anatomy. In addition, it only allows the user to explore the body one part at a time.

To address these problems, in this work, a PC based software system called as, virtual cadaver navigation system (VCNS), has been developed, that can be used to teach human anatomy on real human cadaveric data obtained from the National Library of Medicine

[6]. The system consists of a mannequin, an examination probe (part of 3D motion tracker system), and a personal computer. The segmented and the non-segmented volumetric data that consists of a series of 2D photographic images of the slices of a real human cadaver, forms the input for the software. The system can be operated in tracking, centroid location and labeling modes. In the tracking mode the user can explore the complete virtual human anatomy by obtaining virtual tomographic slices by moving an examination probe over the mannequin. A tomographic slice shows the cross section of the human body at a given orientation and position of an imaginary plane originating from the tracker. The interaction with the mannequin helps in understanding the spatial locations of various organs of the body. In the centroid location mode, the software helps in registering the volumetric data with the the mannequin, in order to obtain correct slices for given position and orientation of the tracker. It is also possible to obtain the tissue specific information by moving the mouse cursor over the desired tissue on the tomographic slice. Thus, VCNS serves as a tool for teaching spatial locations and relative positions of various body parts through the interaction with the mannequin, and augments it with the tissue specific information for enhancing the interpretations. These factors are crucial in teaching human anatomy and are lacking in the existing methods.

VCNS provides real time exploration of very large datasets of complete human body, unlike the previous approaches that allows part of the body to be explored at a given time. This is achieved by dividing the data into small bricks and paging only the required bricks from the hard disk into the main memory and the texture memory for visualization. The software uses only a fixed amount of main memory and the texture memory during the execution, and divides them into slots that are reused to replace old bricks with the new bricks. Spatial and temporal locality is used to speed up the access to the brick data. The system achieves real time frame rates (30 frames/sec and higher) on any personal computer with texture memory as low as 64 Megabytes and main memory of 512 Megabytes. Figure 1.1 shows how the probe can be used to interact with the mannequin to obtain the tomographic slice during the tracking mode.

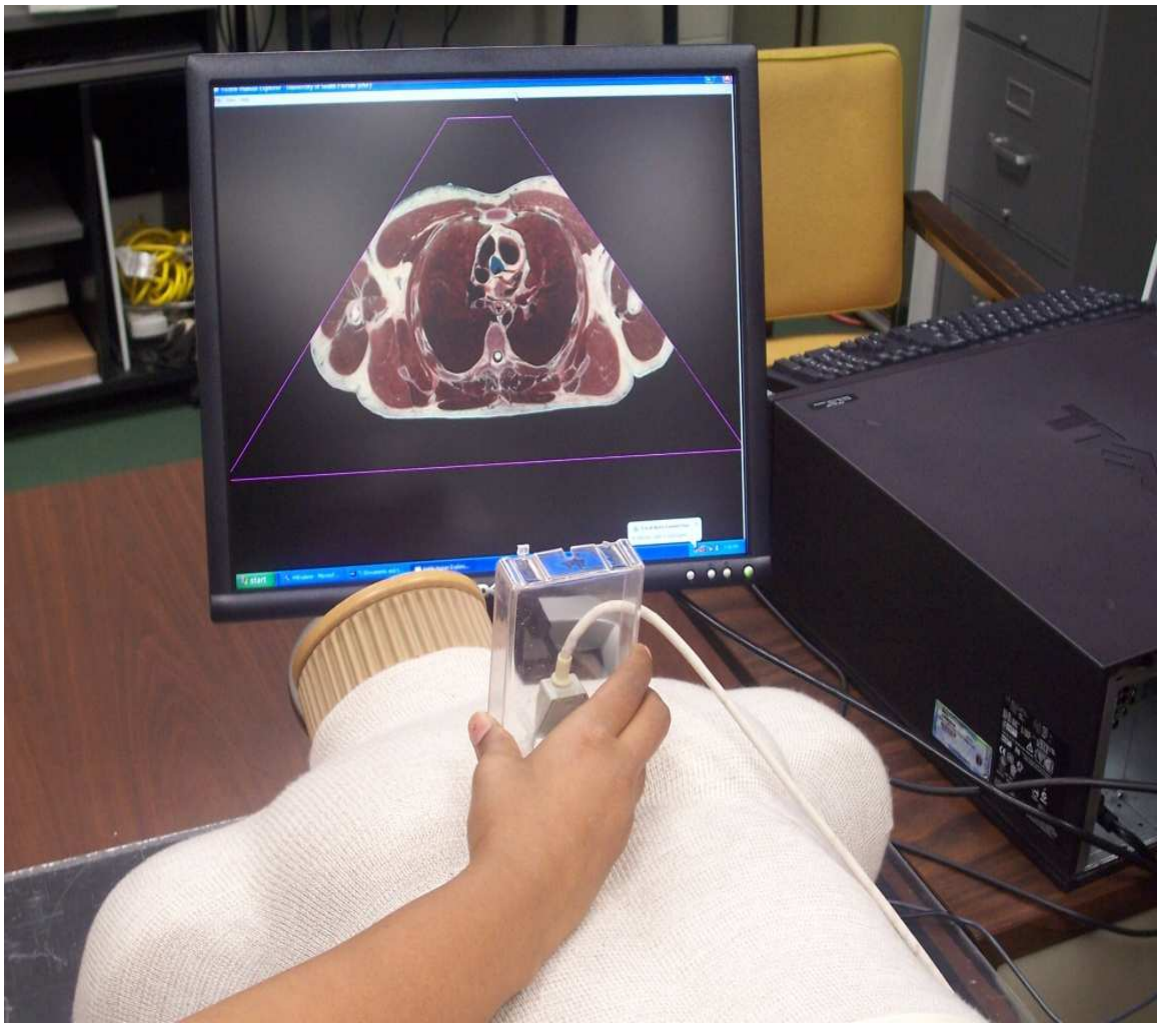


Figure 1.1. The Diagram Shows How VCNS Can Be Used To Obtain Cross Sectional View (Tomographic Slice) Of Virtual Human Body At Arbitrary Locations And Orientation During The Tracking Mode

The following section gives a summary of the contributions of the thesis towards the development of the VCNS.

1.6 Contributions of Thesis

1. A virtual reality framework is developed for learning human anatomy, based on real time navigation and visualization of very large cadaveric data by the way of obtaining virtual tomographic slices, by moving a examination probe over a mannequin.

2. An efficient storage mechanism is developed for organizing and accessing very large volumetric data by dividing it into small blocks called as bricks, and arranging them hierarchically in a 3-D spatial data structure called binary space partition tree (BSP).

3. Algorithms are developed and implemented for:(i) extracting only the required bricks from the BSP tree by performing a 3D collision detection between the tracker plane and the bricks, based on the framework proposed by [13], and (ii) obtaining the points of intersection based on the 3D Sutherland Hodgman clipping algorithm [56].

4. Visualization algorithms are developed for computing normalized texture coordinates from the vertex coordinates of the cutting plane obtained by clipped it against individual bricks, and for mapping 3D texture data from the bricks onto the cutting plane to obtain tomographic slice.

5. Data access algorithms using temporal, spatial and sequential locality are implemented for efficient management of texture memory and the main memory. The least recently used bricks in the texture memory are replaced by bringing the new required bricks from the main memory and from the hard disk into the main memory.

1.7 Organization of Thesis

The thesis is organized as follows. Chapter 2 gives a overview of current research in the area of virtual reality based medical training systems. Chapter 3 explains visible human data set used in the system for navigation. The architecture of the VCNS is presented in

Chapter 4 followed by a detailed design and implementation of the system in Chapter 5. Chapter 6 presents the results obtained under different modes of operation of the VCNS, and a discussion on advantages of using VCNS as a teaching tool and the future work.

CHAPTER 2

LITERATURE REVIEW

Virtual reality (VR) techniques have been proven to be useful in providing medical training in convenient and economic way compared to the conventional training procedures that use dissection of real human cadaver each time. This chapter provides an overview of the current and previous research in the field of virtual reality based training methods in medical science. A review of techniques for visualization of large volume data, and the challenges and issues in using the virtual environments for healthcare, is also presented.

The applications of VR in field of medicine can be classified into following categories [43]:

1. Surgical planning and procedures.
2. Virtual endoscopy.
3. Medical education.
4. Neuropsychological assessment and rehabilitation.

2.1 Surgical Planning and Procedures

Students learning surgical procedures are often trained on inanimate tissues and models due to cost issues. The science of VR opens an entirely new path for acquiring the surgical skills, using computers for training and evaluation. The early efforts focused on creating surgical simulators (for example, the abdominal-surgery simulator by [50], and the limb-trauma simulator by Delp et. al. [16]) suffered from problems such as low-resolution graphics, lack of tactile and force feedbacks and the lack of realistic deformations of organs. In the last decade, efforts have been directed towards developing VR based training systems

to augment the learning experience with more sophisticated and realistic visualizations and sensory feedbacks. For example, the MIST-VR trainer [Ali 02, Gor 03], used for performing basic laparoscopic skills, have been shown to possess more training efficacy than the traditional procedures.

Apart from training surgical students, the use of VR environments in performing remote surgery is opening new possibilities[64]. Using telesurgery, surgeons can participate in the battlefield operations from remote sites; operate on a patient in rural areas, in an airplane, on ship or even at space station, remotely from their office [37]. Besides overcoming the geographical barriers, telemedicine also results in the reduction of exposure to diseases, and reduction in costs as a result of reduced trauma [49]. VR has also helped surgeons by superimposing the real images with the virtual images reconstructed from the MRI and CT data, using a technique known as augmented reality. Such techniques have been shown to be very effective in performing orthopaedic and tumor surgeries [17].

VR has been beneficial in improving the planning process done before the actual surgery. The planning requires the surgeons to mentally integrate a series of two-dimensional MR and/or CT images to form a 3D view of the anatomy. This mental transformation is difficult, since the anatomy is represented in different scanning modalities on separate image series, usually found in different sites/departments [43]. A VR-based system can be used to reconstruct realistic 3D models of the traumatized part with ability to perform grasping, clamping, cutting, and bleeding or leaking of fluids. Thus, VR can improve the way surgeons plan procedures before surgery [37]. One such system is the Netra [23] used for planning precision biopsies, laser-guided tumor resections and surgery for Parkinson's disease. Another example is the Cyberscalpel surgical planning system developed by NASA researchers, which has been successfully demonstrated to plan the operation of the person with cancer of the jaw [46].

2.2 Virtual Endoscopy

Endoscopy involves diagnosis, by inserting invasive or minimally invasive instruments into the patients. These procedures are often not perfect and patients are subject to complications such as bleeding, perforation etc. Also, the cost of endoscopy is significant. As a result, virtual endoscopy is being explored as an alternative by different researchers [48, 26]. It involves performing standard CT or MRI scans of the area of body of concern and segmenting the various organs and tissues. Sophisticated flight path algorithms, derived from terrain tracking algorithms used in the military, are used to fly through these organs providing images comparable to performing real video endoscopy [35, 51]. This technique can be extended to explore the parts that are inaccessible to the real endoscopic instruments, either because it is too dangerous (such as parts inside the eye) or too small (inner ear) [49]. Typical examples include colon, stomach, esophagus, tracheo-bronchial tree, sinus bladder, ureter and kidneys, pancreas or biliary tree [38]. Virtual endoscopic simulators are cost effective and completely non-invasive with no complications to the patient unlike their real world counterpart [19].

2.3 Medical Education

VR provides a deeper understanding and appreciation of the anatomical structure and relationships by allowing the learner to fly around and/or through the various organs. The extraordinary perspectives made available by these learning tools can prove more effective than the any of the other means, including cadaver dissection. Anatomy teaching being mainly descriptive can be greatly benefited by VR environments [18]. Several web based applications have been developed based on the data obtained from the visible human dataset [55] to teach the anatomy by obtaining cross-sections through the virtual human [6]. The availability of the visible human data over the internet under a no-cost license agreement has spurred the creation of huge number of educational virtual environments.

Due to shortage of cadavers required for dissections, a 3D lifelike detailed virtual human body offers a workable replacement. In fact, these 3D models can allow more realistic

training than the use of cadavers. For example, it is possible to simulate blood flowing out of a virtual blood vessel when it is pricked unlike the cadavers, in which the color is changed and the arteries no longer pulsate [37]. Also, previously cut parts can not be reattached. In the virtual environment the procedures can be repeated infinitely, and can also be stored for analysis for the later group.

Thus, VR offers dynamic environment in which models of various organs and systems can move during normal or diseased states, or respond to various externally applied forces providing an experimental and didactic platform for learning human anatomy.

2.4 Neuropsychological Rehabilitation and Psychology

VR is emerging as a promising medium for treating patients with psychological disorders. It is ranked 3rd out of 38 psychotherapy interventions, that are predicted to increase in the next 10 years [40]. In the field of psychology, VR has been used to create real world situations tailored according to the patient's psychological disturbances. In the virtual environments, nothing the patients fear can really happen to them, giving an assurance that they can freely explore, experiment and experience feelings and thoughts [14]. This provides them not only an awareness to do something to create change in the environment they are immersed in, but also to experience a greater sense of personal efficacy [43]. Till now, the clinical effectiveness of VR has been verified in the treatment of six psychological disorders: acrophobia [20], spider phobia [21], panic disorders with agrophobia [60], eating disorders [44] and fear of flying [47, 36, 63]. However, most of this research is based on controlled studies and pilot trials, with limited convincing evidence about the efficacy and practicality of their use.

Although, VR has been used in the area of cognitive rehabilitation [52, 45], there are no clinical trials to support their efficacy except in the assessment of cognitive functions in persons with acquired brain injuries [43]. VR has been shown as a very effective psychometric tool in this field [65, 41].

2.5 Issues and Challenges in using VR for Healthcare

There are concerns about using VR in clinical environments. The safety issues that arise in using VR systems are symptoms such as motion sickness, strain on ocular system, reduced sense of presence, and development of responses inappropriate for the real world, which might lead to negative training [33].

These symptoms can be reduced with improved quality of VR simulations. As noted by [42, 39], in most of the individuals these effects are transient and minor, and subside quickly. Nonetheless, precautions should be taken to ensure safety of patients by monitoring and controlling their exposure to virtual reality environments [33].

The technical issues include the lack of standardization in the systems, and the performance factors. Currently, every VR system requires dealing with conflicting hardware and driver software and typically requires a dedicated staff or a computer technician to ensure its smooth working. Other technical challenges in implementing VR systems include these [25]:

1. High computation needs of virtual acoustic displays for simulating even a small number of sources.

2. Limited functionality of tactile feedback mechanisms.

3. Quality and performance tradeoffs offered by image generators that can not provide low display latency.

4. Inadequate robustness, small working volumes, latency and poor registration of position trackers.

5. Limited field of views and encumbering form factors of Head Mounted Displays.

6. Lack of effective olfactory sensors.

Finally, the high costs associated with the development and management of the VR systems is also a major issue in widespread usage of the VR based systems. The cost

required for designing a clinical VR application from scratch, and testing it on clinical patient may range between 150, 000 and 200, 000 US dollars [43].

2.6 Visualization of Large Volume Data

Although different sensory feedbacks are vital to virtual realism, visual feedback still remains the most important form of feedback in the VR based systems. With the advent of new imaging techniques such as Magnetic Resonance Imaging (MRI), Computed Tomography (CT), Positron Emission Technology (PET) the amount of data that the clinicians need to handle is increasing. In order to provide realistic visualizations, it is necessary to present the output with finest details possible in real time. Thus, it is necessary to manage the large high resolution volume data required for 3D rendering. This section provides an overview of the research in the visualization and large data management techniques.

Volume visualization is a classical problem in computer graphics. The volume data can be either surface rendered [34] or volume rendered. In case of volume rendering, it is possible to see inside the surface, making it useful for medical visualization of organs. The fastest software based volume rendering algorithm is the "Shear-Warp Factorization", which operates by factoring the view transformation into a 3D shear parallel to the data slices, a projection to form an intermediate but distorted image and a 2D warp to form an undistorted final image [31]. New techniques combine hardware support for 3D textures [22] and multi-resolution techniques to display the volume on computers with low texture memory and at real time frame rates.

The 3D texture support from the graphics hardware has made real time rendering possible. Still the size of the data remains a problem because of the limited amount of texture memory available on the graphics boards. The multi-resolution method with texture paging provides a solution to the data size problem by compromising the quality to a tolerable extent. It operates by dividing the volume data into small data bricks and arranging them with decreasing level of resolutions using hierarchical data structures [Plate et. al. 2002]. In this approach, a particular resolution is selected for the brick,

depending on its distance from the camera. The bricks closer to the camera are displayed at the higher resolution than the one farther from the camera. This technique makes visualization of data as large as 16GBytes feasible on computers with texture memory as low as 64 Mbytes. For example, LaMar et. al. [32] have used the multi-resolution hardware based texture rendering for large volume visualization and viewing arbitrarily oriented slices through Visible Female dataset, while Volz [62] have used combined hardware and software techniques for viewing seismic datasets.

2.7 Context of this Work

Most of the previous work in the area of VR based medical training is focused on surgical training and simulations. There are only a few systems specifically devoted to teaching anatomy [57] [18]. The main drawback of these systems is that, they are targeted at specific parts of the body, and lack integration of high level tissue specific information, which is imperative for learning human anatomy. In addition, they offer limitations in terms of the amount of data that can be explored at a time. The work presented in this thesis is aimed at combining the advantages offered by the previous techniques with a high level knowledge of tissue related information thereby providing an a comprehensive for anatomy learning system. In addition, the problem with the data management limiting the previous systems is overcome by intelligent texture paging, and hardware based 3D texture mapping techniques. Thus, the proposed system offers a cost effective comprehensive teaching tool, which can be used for teaching anatomy as well as for self learning.

CHAPTER 3

VISIBLE HUMAN DATASET

The system has been tested on the data obtained from from the National Library of Medicine's (NLM) Visible Human Project. The aim of this project was to obtain MRI, CT and high resolution RGB images of the human male and female. The following sections explain the technique used for acquiring the data sets and the file format of the data.

3.1 Visible Male

NLM awarded a contract to the University of Colorado Health Sciences Center to create the digital cross-sections of a 39-year old convicted murderer (male) who had donated his body to science. MR and CT data were captured from the unfrozen specimen just a few hours after the death to avoid the deterioration. The MR scans of the head were taken along the axial plane, while the remaining body scan was performed along the coronal plane. The slice thickness was 4 mm in both the cases. The images were stored as 256 X 256 X 16 bit in the General Electric Genesis format [55]. The CT scans from head to neck were taken at 1mm intervals, every 3mm through thorax, abdomen and pelvis, and every 5mm in the lower extremities [55]. After acquiring the MR and CT, the male cadaver was frozen by placing it in a specially constructed chamber covered with dry ice and was divided into four sections from head to toe. The first section was from leg to toes, second included knees and thighs, third was made of abdomen and pelvis and the fourth contained head, neck and thorax. Each section resulted in a block which was milled at 1mm interval using a cryomacrotome (cutting machine) developed at University of Colorado Medical School. As each layer was exposed, a color RGB photograph was taken. This process captured the images in 2048 X 2048 X 14 bit TIFF format. This raw data was further processed to

obtain 1800, 2048 X 1216 X 24 bit images at 1mm intervals resulting in a RGB data of size 9GB. The data can be downloaded from the NLM website [6]. Figure 3.1 shows the MRI, CT and RGB slice of the visible human head.

3.2 Visible Female

The Visible Female images were obtained from a 59 year-old woman who died of coronary artery disease. Although, the process was similar to that of the Visible Male, the slices were obtained at 0.33 mm regular intervals as opposed to the 1mm thickness. The Visible Female contains slightly over 5,000 images with a total of 39 GB. Figure 3.3 shows the RGB slice of the visible female head.

3.3 Segmented Dataset

The segmented VH male dataset was created at University of Michigan, by manually labeling every voxel in the 16-bit gray scale images obtained from the original colored VH male data. The 16-bit gray scale values are converted to a 24 bit colored segmented image by assigning unique R, G and B color values to every class. There are approximately 1600 structures classified and labeled and can be loaded into the VCNS using the name look up file. This file consists of comma separated R, G, B values and their corresponding class name. A transaxial slice through the segmented volume is shown in Figure 3.2.

3.4 Data Format

Currently, VCNS supports only the "multiple directory tagged image file format (TIFF)", that can store multiple TIFF images within a single TIFF file. However, it is possible to load the data in different formats directly into the system by implementing a single method (*loadDataBrick*) of the plug-in class called *VolumeDataLoader*. The TIFF image is defined as a sequence of bytes that starts with a file header called as *image file directory* (IFD) and pointers to the actual image data from the the IFD. In case of the multiple directory TIFF file there are more than one IFDs per file. The TIFF file format is explained in detail

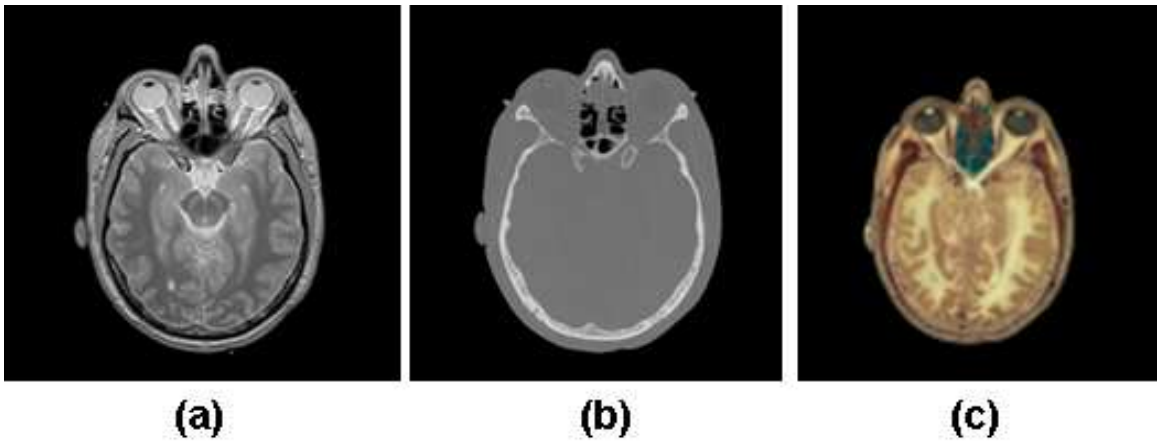


Figure 3.1. Visible Human Male Images (a) MRI Image (b) CT Image (c) High Resolution Colored Image

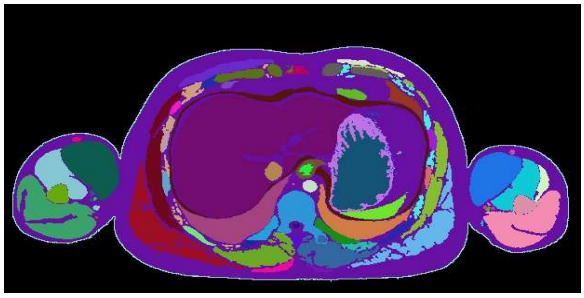


Figure 3.2. Segmented Transaxial Slice Through Visible Male Thorax

Figure 3.3. A Transaxial Slice Through Visible Female Head

in [8]. VCNS uses a software library called "libtiff" [3] to read and write the TIFF files. This library also provides functions for converting the TIFF files into several other formats such as JPEG, PostScript etc.

When the raw data is loaded as a TIFF file into the system, the data is converted into bricks and written to a file called as "brick file", that stores the data corresponding to a brick contiguously. This scheme of contiguous data organization helps in reducing the download time for the brick data, from the hard disk. The brick file is generated for the raw input data as a part of preprocessing step. This file can be created once, for a given input data, and can be used subsequently. This speeds up the startup time for the software since the preprocessing step of bricking is skipped. The format of the brick file has been designed specifically for VCNS is as follows:

//Fileheader

Magic Number : 1 byte (must be 0XBF)

data format : 2 bytes

number of components per voxel : 1 byte

number of bytes per component : 1 byte

voxel dimensions in mm: 3 bytes

number of rows: 4 bytes

number of columns: 4 bytes

number of bricks: 4 bytes

brick dimensions: 4 bytes

size of bricks in bytes: 8 bytes

//Actual Brick Data .....

CHAPTER 4

ARCHITECTURE OF VCNS

This chapter describes the software architecture of the proposed system. The various modules comprising the system are described first, followed by the computational flow of the system as well as that of individual modules. Finally, the hardware components used in the system are described along with the graphical user interface (GUI) designed to interact with the system.

4.1 Modes of Operation

The system is designed to operate in three different modes which are as follows:

1. Tracking mode: In this mode, the real human cadaveric data can be explored by moving the examination probe over the mannequin. The software computes the 2D planar slices from the data in real time and the output image is shown on the display in a form similar to the ultrasound image.

2. Centroid Location or Registration mode: This mode helps the user to align the centroids of the mannequin, and the volumetric data, loaded in the software. It is necessary to align the centers before exploring the data, in order to obtain correct slices in the tracking and labeling mode.

3. Labeling mode: This mode is used for exploring the data for which, the segmentation information is available. It works in the same way as the tracking mode except that, the resulting image can be used to obtain tissue specific information, such as its name. This mode can not be used if the segmentation data is not available.

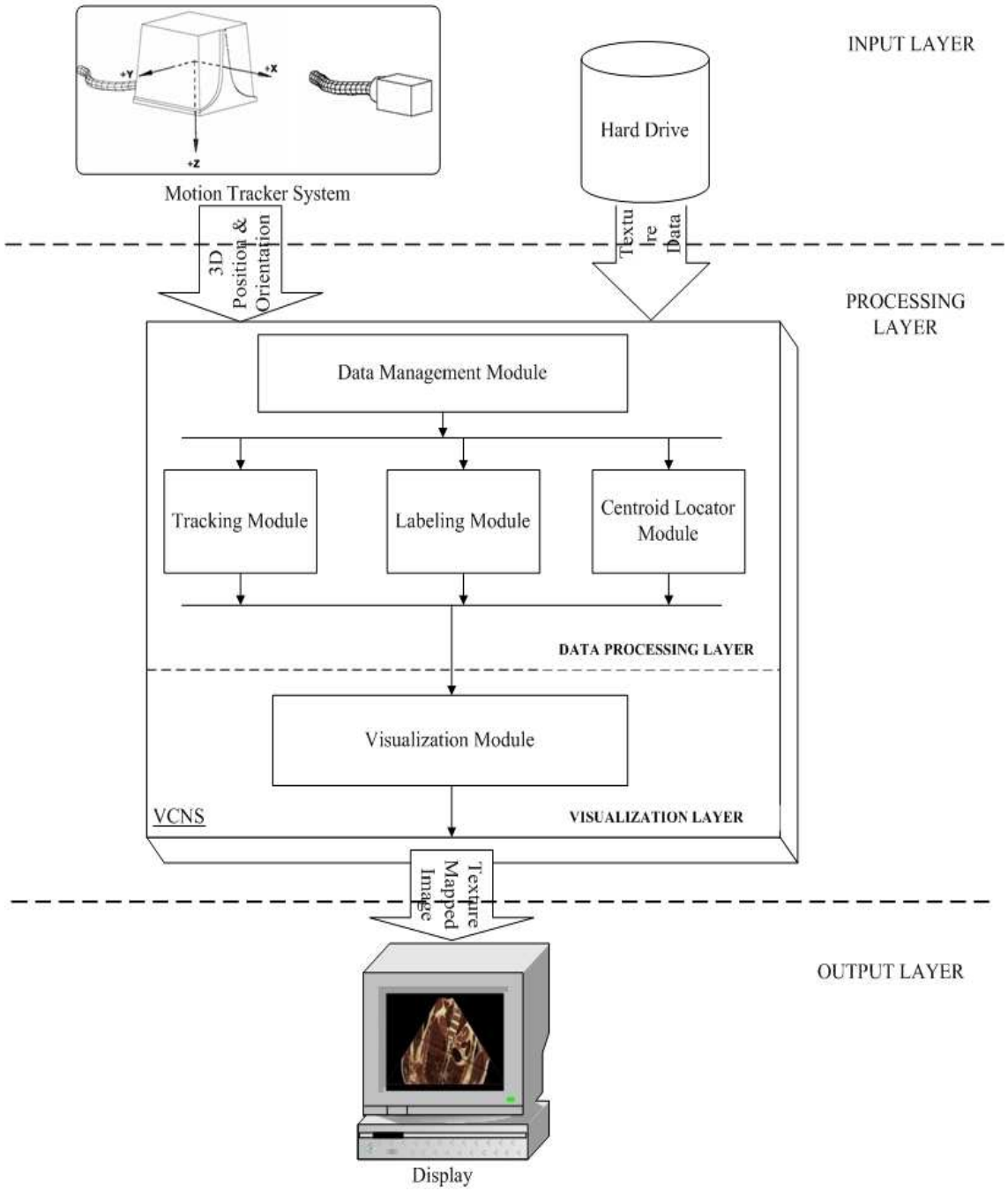


Figure 4.1. Software Architecture Of VCNS Showing Different Modules, And Inputs And Outputs Of The System

4.2 Software Architecture

In order to perform the above functions, the software is divided into several modules. Figure 4.1 shows the architecture of the complete system. The inputs to the system are comprised of the segmented volume data and/or the non segmented volume data. Volume data consists of series of tomographic images acquired using techniques such as magnetic resonance imaging (MRI), computed tomography (CT) or obtained from the visible human dataset. When the VCNS is operated in one of the above execution modes, it performs various data management steps before obtaining the final output. This functionality is divided into different modules, which are as follows:

4.2.1 Data Management Module

This module is responsible for handling all the data related issues, which include data organization and data access. It is capable of performing management of very large data, by dividing it into small blocks called as "bricks", that represent a subvolume of the original volumetric data. The process of forming the bricks from the original raw data is called as "bricking". It is required to convert the raw volumetric data into the bricked form before it can be used in the system. This enables faster access and downloading of a brick from the hard disk since all data corresponding to a brick is written contiguously on the disk. The functionality of this module is divided into the following submodules as shown in the Figure 4.2:

4.2.1.1 Data I/O

This module handles the reading and creation of volumetric bricks from the raw data. It is made up of two parts: (i) Data Loaders, that are responsible for reading the brick data as well as the raw data from the hard disk, and can be customized to support different file formats, (ii) Data Writers, that are used to write the bricked data generated as a result of the bricking process. The data writers augment the raw data with the transparency values for every voxel during the bricking process, so that it can be used as a texture using

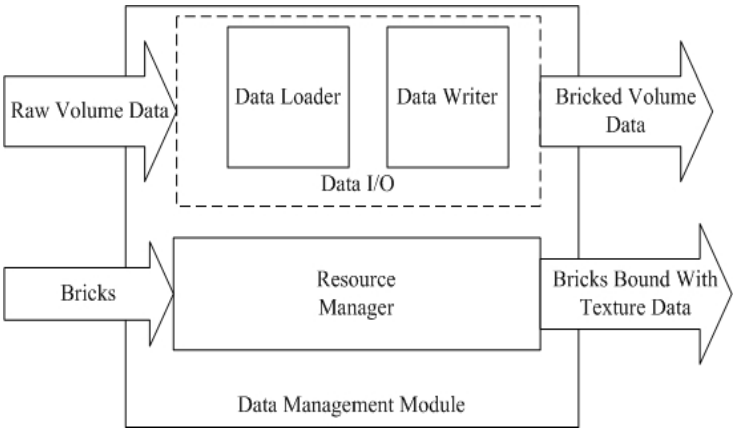


Figure 4.2. Software Architecture Of Data Management Module

OpenGL texture mapping routines. Thus, the data generated as a result of the bricking is bigger in size, than the raw input data.

4.2.1.2 Resource Manager

This module is the most critical part of the system, and is responsible for allocating, accessing, and managing the main memory and the texture memory resources at runtime. It allocates a fixed amount of the main memory and the texture memory, and divides them into slots of size that is a multiple of brick data size. It maintains a direct mapping between the slot addresses and the brick addresses, and implements a tagging scheme in order to locate them in the memory. This module also implements a LRU (least recently used) replacement policy, to replace the bricks that are no longer needed, and uses the spatial and temporal locality to minimize the access time for the brick data for the required bricks.

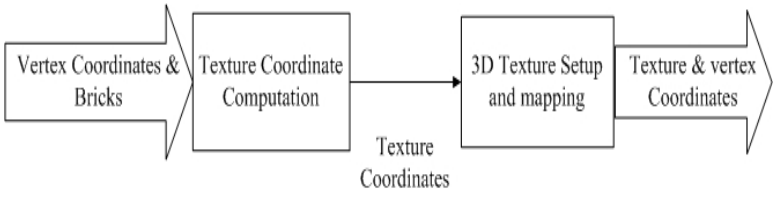


Figure 4.3. Software Architecture Of Visualization Module

4.2.2 Visualization Module

The visualization module implements the algorithms for computing the texture coordinates and 3D texture mapping. 3D textures consists of a stack of two dimensional texture images and have to be loaded into the texture memory before they can be used for texture mapping. 3D texture mapping is a technique of extracting a part of the data at arbitrary positions and orientations from this image stack, and requires interpolating the texture data at the points where it is not available. This is usually implemented by the graphics hardware using trilinear or higher degree interpolation. The visualization module uses the hardware support for 3D textures, in order to extract the slice from the volume data. In order to achieve this, it is necessary to specify the texture coordinates for the slice to be mapped, in a normalized form. The inputs to the visualization module include, the texture data for the bricks, and the coordinates of points of intersection of the cutting plane with that of the bricks. The function of this module is divided into two parts (Figure 4.3), which are, texture coordinate generation, and 3D texture mapping. The output of this module is the set of vertices and their corresponding texture coordinates that are sent through the OpenGL graphics pipeline, to obtain the final texture mapped image.

4.2.3 Tracking Module

The functionality of the tracking mode is encapsulated in this module. It is made up of several other modules as shown in the Figure 4.4. This module handles the data flow between the various modules, in order to obtain the final texture mapped image in the tracking mode. For example, it computes the geometry of the virtual cutting plane

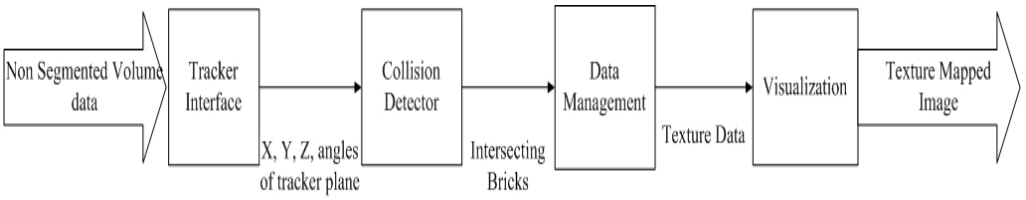


Figure 4.4. Software Architecture Of Tracking Module

emanating from the probe, queries the collision detector module to find out the bricks intersected by the plane at a given time, requests the texture data for these bricks from the data management module, and finally provides the data necessary for visualization of the texture mapped image to the visualization module.

4.2.4 Centroid Locator Module

This module encapsulates the functionality of the centroid locator mode. It is made up of several submodules as shown in the Figure 4.5 and coordinates the data flow between these modules. When the system is operating in the centroid locator mode, this module queries the positions of the three orthogonal planes from the control panel dialog using the plane position tracker interface, and computes their geometry based on this information. It stores the bricks intersected by each of the selected orthogonal planes and requests the texture data corresponding to them from the data management module. All the selected bricks are sent to the visualization module to obtain the final image.

4.2.5 Labeling Module

The segmentation and tissue classification information are essential inputs for this module in addition to the non segmented data. The tissue classification information is provided as a mapping between their anatomical names and color value of the class representing the tissue. The functionality of this module is similar to that of tracking module with a difference that, it uses the label generator module for obtaining the tissue specific classification information for the resulting slice. In order to obtain the labels, this module generates the tomographic slice from the segmented as well as the non segmented dataset in the way sim-

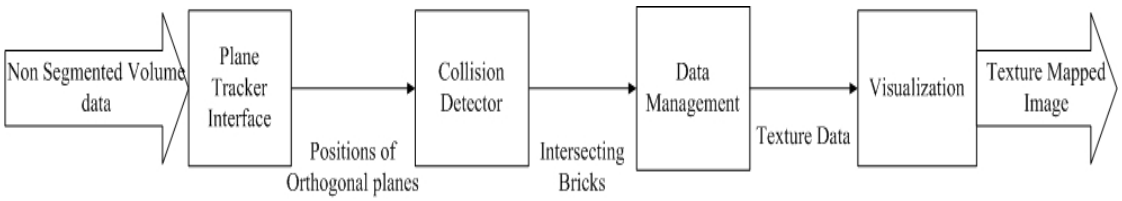


Figure 4.5. Software Architecture Of Centroid Locator Module

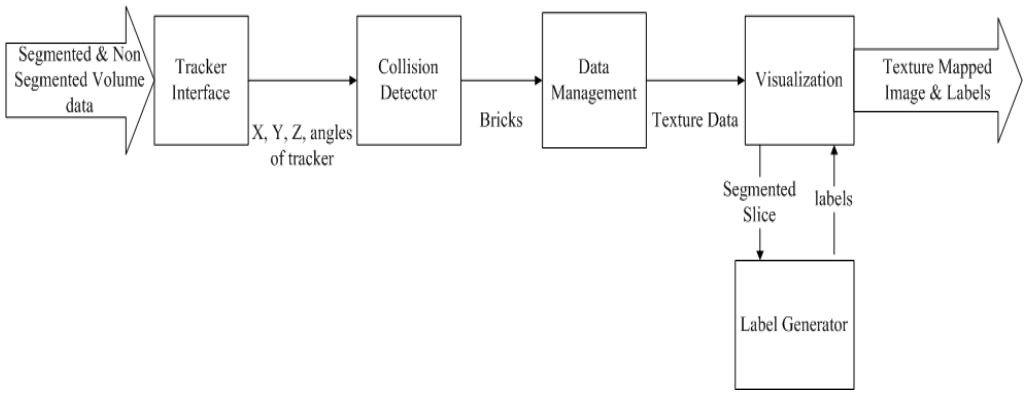


Figure 4.6. Software Architecture Of Labeling Module

ilar to the tracking mode. This information along with the tissue classification information is given as input to the label generator module to obtain the labels for the tissues on the slice. The various submodules comprising this module are as shown in the Figure 4.6.

4.2.6 Collision Detection Module

This module is used by the tracking, centroid locator and labeling module for determining the bricks that are intersected by the plane at any given time. The input to this module is either the cutting plane emanating from the probe (in case of tracker and labeling mode) or one of the three orthogonal planes (in case of centroid locator mode). This module defines a binary space partition (BSP) tree for arranging the bricks comprising the volume data. A BSP tree (Figure 4.7) is a spatial data structure that organizes the volume data, by recursively dividing it into smaller subregions, and arranging them in a binary tree structure. The root of the tree represents the whole volume, internal nodes represent the subregions obtained by dividing the region corresponding to the root, along one of the three x, y or z planes, and the leaf nodes represent the bricks. The collision detection module uses the 3D collision detection algorithms proposed by [13] to successively query the internal nodes to determine intersection with the cutting plane till it reaches the leaf nodes and returns the bricks intersected by the plane as shown in the Figure 4.8. It is also determines the points of intersection of the plane with the each of the intersecting bricks

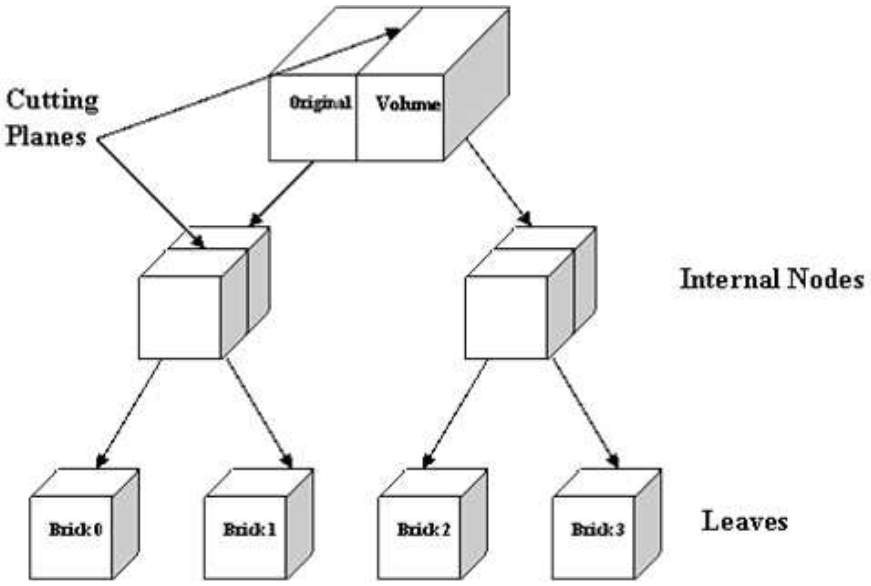


Figure 4.7. Binary Space Partition Tree

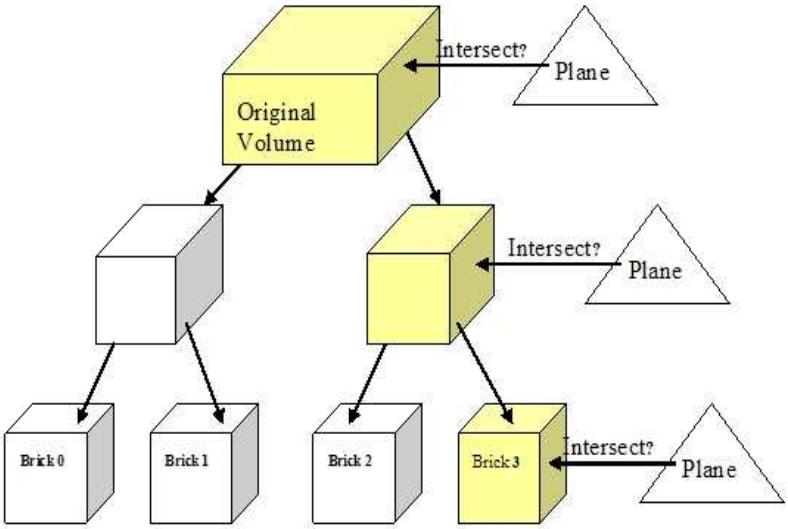


Figure 4.8. Collision Detection Process

by clipping the plane against each brick using Sutherland Hodgman 3D polygon clipping algorithm [56].

The other modules used in the system are, tracker interface module used by the tracking module to initialize the motion tracker system and track the position of the probe to obtain its 3D position and orientation, and the plane tracker interface module that is used in the centroid locator mode to obtain the positions of the plane from the control panel dialog box.

The following section gives the computational flow of the overall system, and explains in detail, the steps performed in each of the modes, and the resulting data flow across various modules.

4.3 Data and Computational Flow of VCNS

The flowchart summarizing the computational flow of the complete system is shown in the Figure 4.9. Initially, the system is in the idle state waiting for the input from the user. When the user loads the raw non segmented volume data, it is first converted into bricks and written to an intermediate file as a part of preprocessing step. Next, the appropriate execution path is selected depending upon the current mode of operation. In case of labeling mode, the user is prompted for segmentation and tissue classification information before execution.

The following sections explain the computational flow and the data flow within the individual operating modes.

4.3.1 Tracking Mode

In this mode, the probe can be moved over the mannequin to obtain virtual tomographic slices from the volume data. The tracking module reads the position and orientation of the probe and generates the plane based on this information. Next, the plane geometry (3D coordinates of the vertices) is passed to the collision detector module to obtain the intersecting bricks and the points of intersection as shown in the Figure 4.10. The data

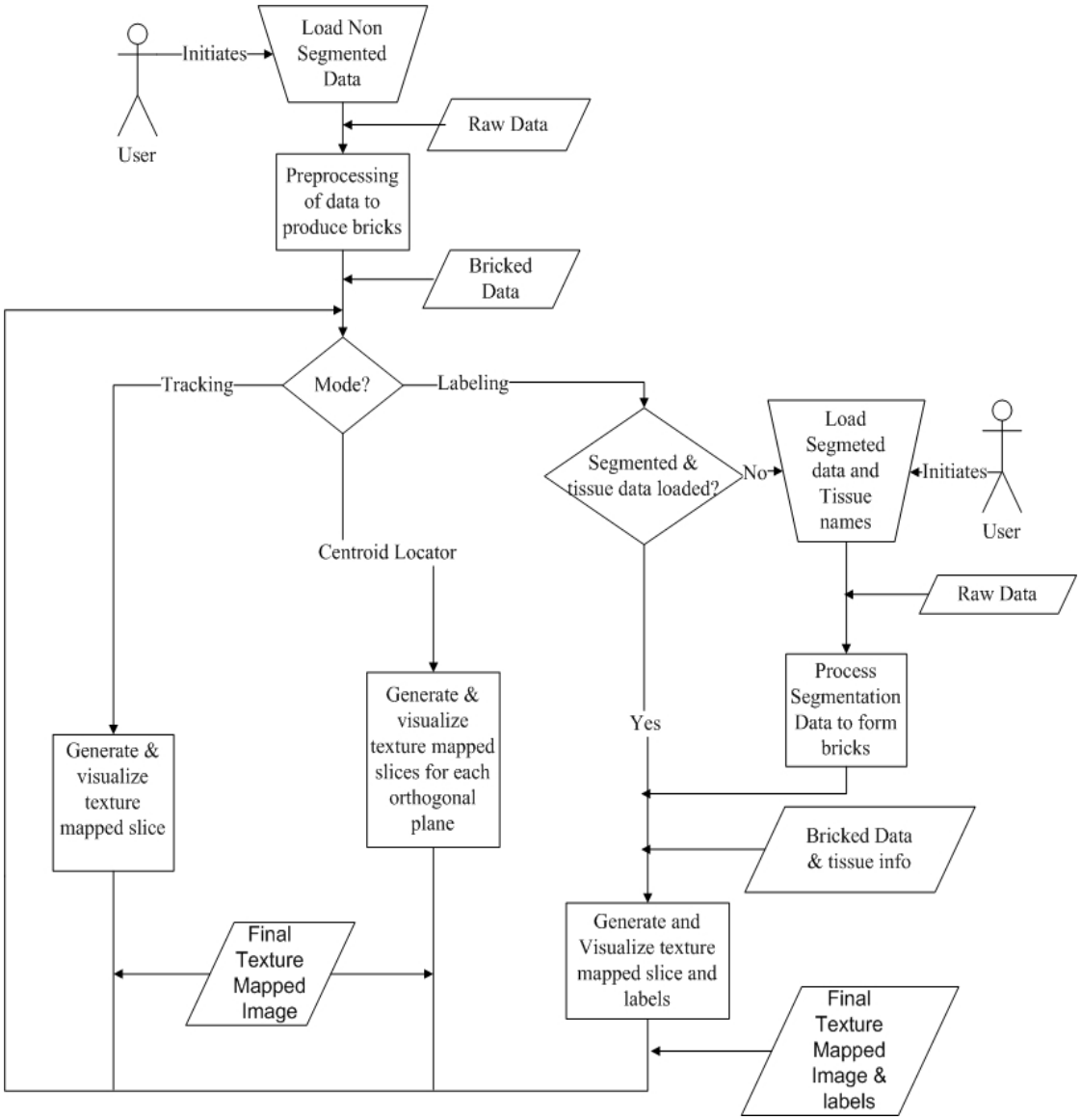


Figure 4.9. Computational Flow Of VCNS

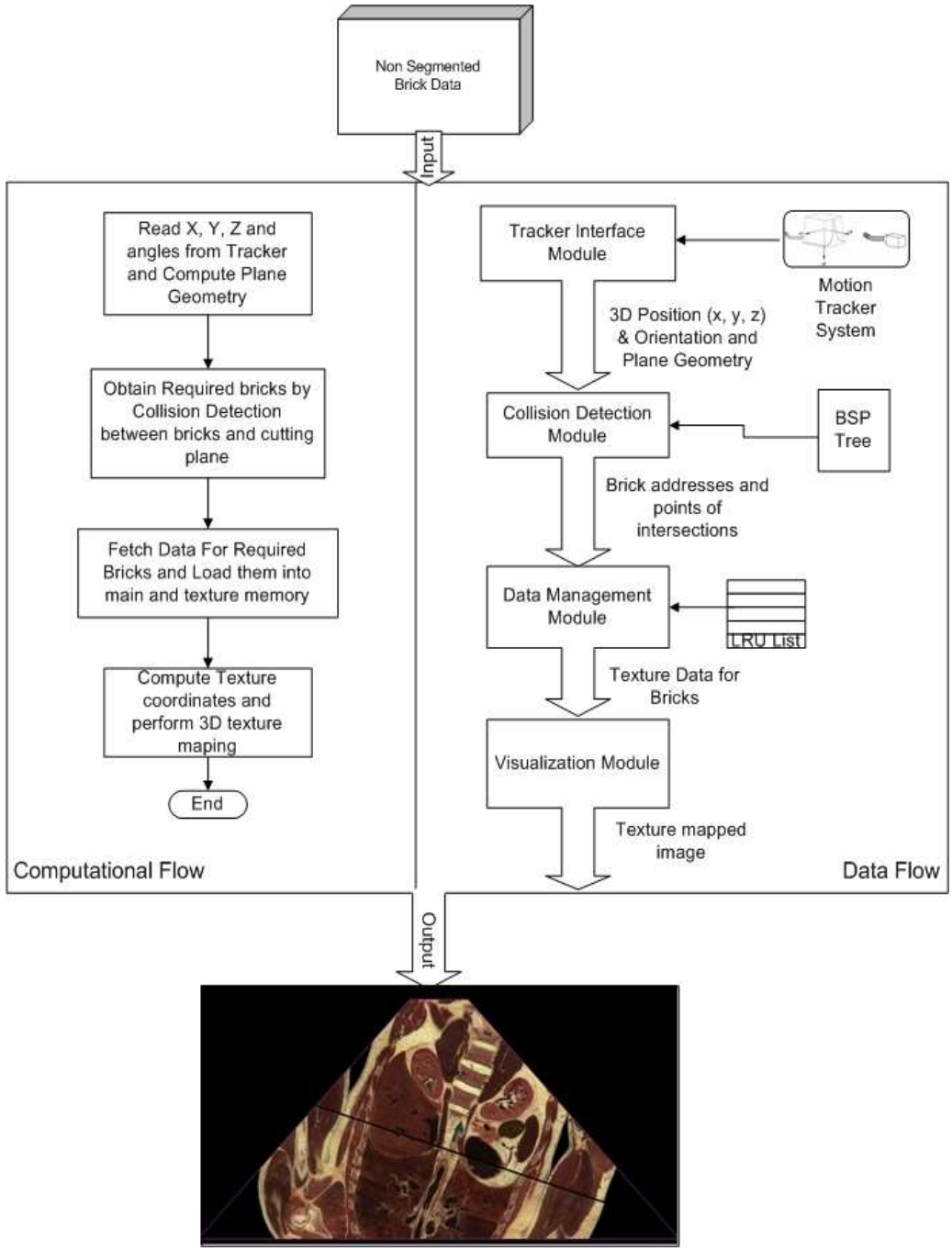


Figure 4.10. Computational Flow Chart And Data Flow Diagram For Tracking Mode Of VCNS

management module arranges the the bricks such that bricks already present in the texture memory are processed before the bricks residing in the main memory, followed by those present on the hard disk. If the resulting number of bricks are more than the maximum number of bricks that can be accommodated in the texture memory, the tracking module processes them in batches. Each batch is passed to the data management module for texture binding. The intersection data and the texture data for the selected bricks is finally passed to the visualization module as shown in the Figure 4.10 (b). The computational flowchart of the tracking algorithm is shown in the Figure 4.10. The input to the tracking mode is the non segmented bricked data and the output is an image similar to an ultrasound image as shown in Figure 4.10.

4.3.2 Centroid Locator Mode

The system is operated in this mode to align the center of the mannequin and the center of the 3D motion tracker system with that of the volume data. This is done by moving the three orthogonal planes, called as transaxial, sagittal and coronal, using the slider controls on the control panel dialog box shown in the Figure 4.14. The point of intersection of the three slices is considered as center of the volume and is used for making corrections to the position of the probe in the tracking mode. The computational flowchart for this mode is shown in the Figure 4.11. The procedure for obtaining the texture mapped image is same as the tracking mode, and is repeated for each of the selected orthogonal planes. The input to the system is the non segmented bricked volume data. The data flow, and the output of the system are shown in the Figure 4.11.

4.3.3 Labeling Mode

In this mode, the segmentation data is used to obtain tissue specific information from the tomographic slice. In order to obtain the correct classification information, the cutting plane emanating from the examination probe is used to obtain the slice from segmentation data, in addition to the non segmented data slice. The procedure for obtaining the slice is same as the tracking mode, as shown in the Figure 4.12. In this case, however, the

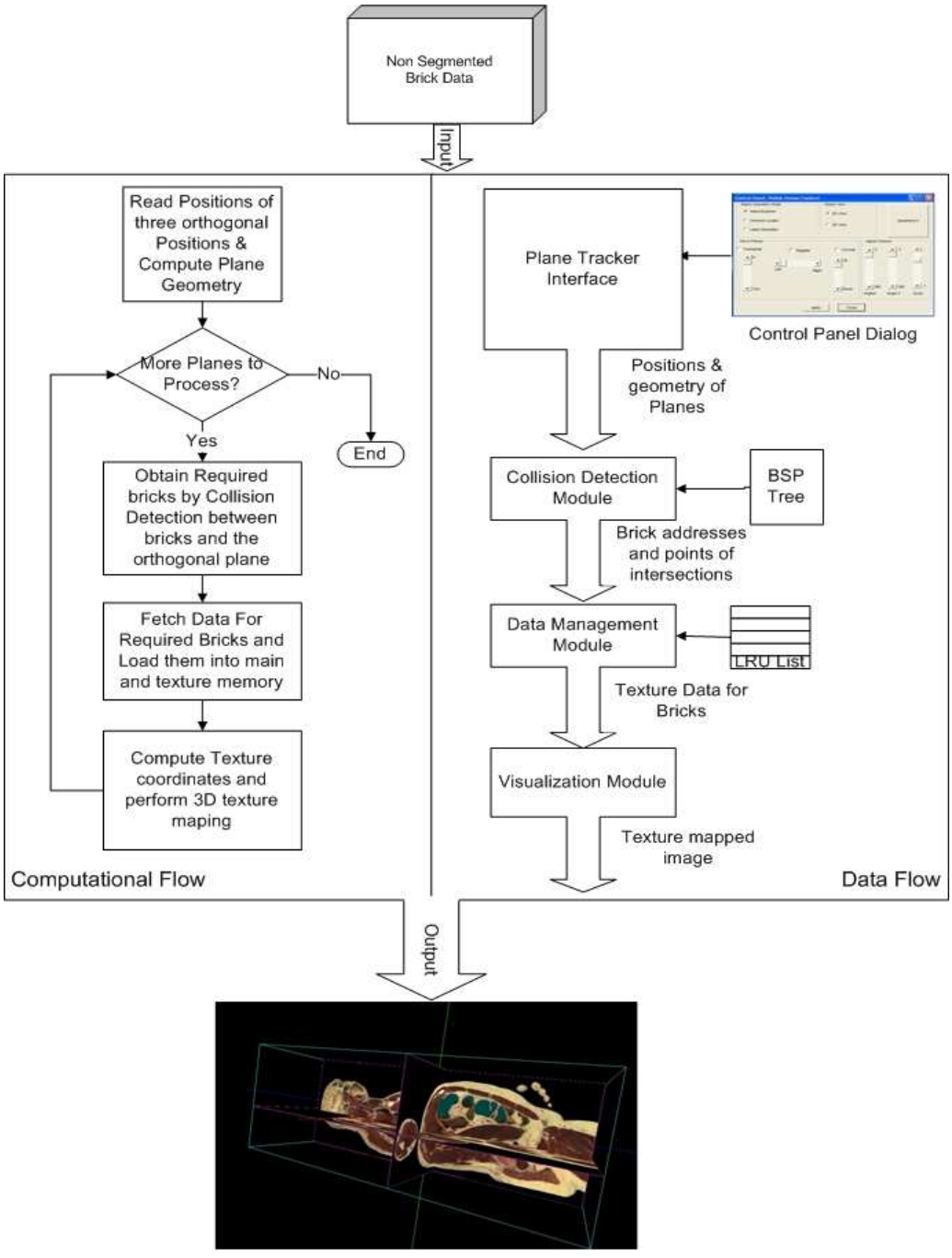


Figure 4.11. Computational Flow Chart And Data Flow Diagram For Centroid Mode Of VCNS

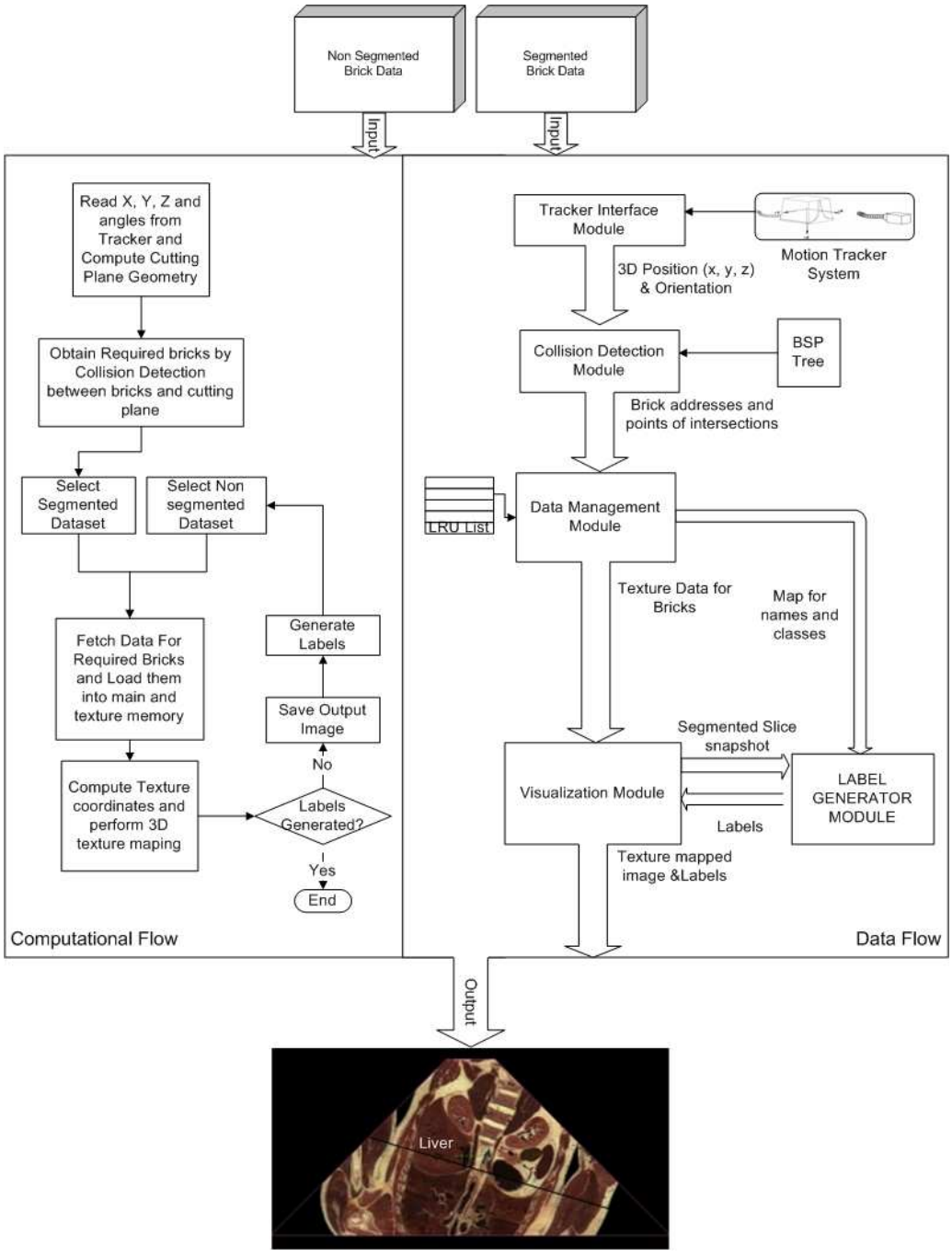


Figure 4.12. Computational Flow Chart And Data Flow Diagram For Labeling Mode of VCNS

output of the visualization is saved after obtaining the slice from the segmented dataset and is further processed obtain the labels by the label generation module. The labels are determined for all the tissues present in the slice by scanning the segmented slice obtained in the above process, and are overlaid on the slice obtained from the non segmented data. Labels can also be obtained for a single tissue, by right clicking on a voxel on the resulting slice as shown in the Figure 4.12.

4.4 Hardware Components of the System

Following hardware components are used in the system:

1. pciBird^{TM} Motion Tracker System.
2. Graphics Workstation.
3. Mannequin.

4.4.1 PCI Bird Motion Tracker

The pciBird^{TM} is an electromagnetic 3D motion tracker system developed by Ascension Technology [2]. It is made up of following three components:

4.4.1.1 Transmitter

A transmitter consists of a high permeability core with three sets of windings (X, Y, Z) placed at right angles to each other and produce magnetic fields, when current is passed through them [11]. The strength of the magnetic field is highest near the transmitter, and decreases with inverse of the square of the distance from the transmitter. Figure 4.13 (a) shows the transmitter and the position of the cartesian coordinate system inside it.

4.4.1.2 Sensor

A sensor (Figure 4.13 (b)) is a precise 3-axis ring core fluxgate magnetometer with a high permeability core at the center. It operates by periodically driving the core to saturation

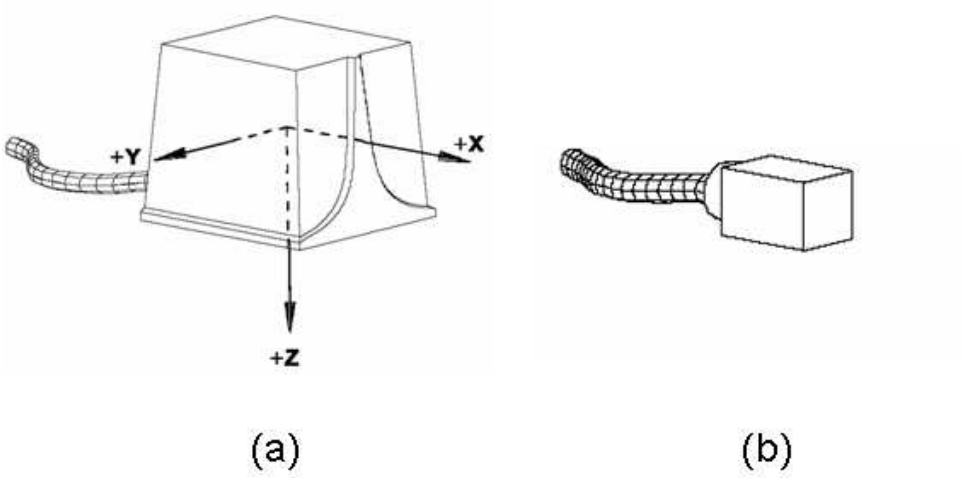


Figure 4.13. Reference Frames For (a) Standard Transmitter (b) 25 mm sensor (Courtesy: Ascension Technology)

making its permeability close to that of the air. The fluxgate magnatometer, then, changes the coefficient of magnetic permeability (μ, where B = μ H) to 2 and measures the EMF at the ends of the coils, which is proportional to the DC magnetic field near the sensor. Ring core fluxgates offer very high performance as compared to the other sensing technologies [11].

4.4.1.3 Electronics

The electronics consists of a full length (12.23 inch) 32 bit v2.1 PCI card and handles transmitter drive circuitry, sensor signal processing, data conversion, processing, power conditioning, and host interface functions [11].

4.4.1.4 Measurement Technique

The tracker system measures the position (X, Y, Z) and orientation (Pitch, Roll, Yaw) of the sensor in every measurement cycle. The number of measurements taken per second can be programmed using the software interface. A measurement is taken by successively energizing each of the three coils of the transmitter to the point of stability. At this point, the current induced at the coil along corresponding axis of the sensor is measured,

and transmitted to the electronics. The electronics calibrates the received data, subtracts noise, and computes the angle and position. The final output is made available at the host interface and can be read using the software interface. It is possible to measure the position data in different formats such as integer, float, and the orientation using Euler or Quaternion methods.

The specifications of workstation are shown in the Table 4.1. The dimensions of the mannequin must be as as that of the subject used for the volume data.

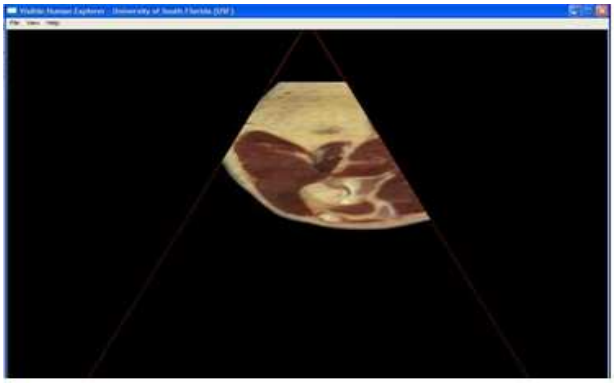
Table 4.1. Specifications Of Workstation

Workstation	
CPU	Dual Intel P4 3.6 GHz With Hyperthreading support
Memory	4GByte 533 Mhz
Graphics Card	NVIDIA Quadro FX 4400 512 MB
Hard Disk	73GB Seagate Ultra SCSI 15000 RPM Non-RAID

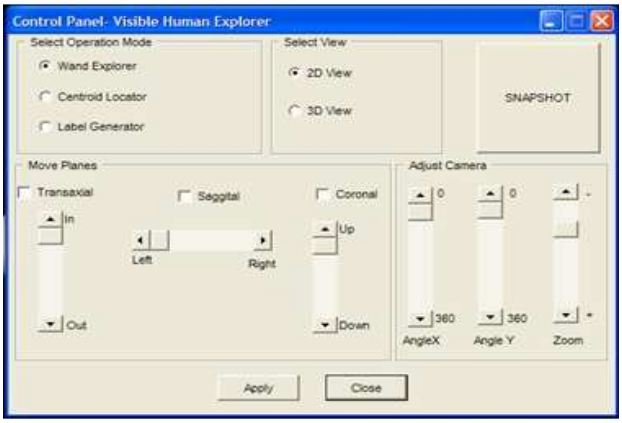
4.5 Graphical User Interface

The graphical user interface (GUI) has been developed using Microsoft Foundation Class (MFC) library under the Visual C++ development environment. There are two windows used for interaction. The main window (Figure 4.14 (a)) is used to show the outputs from various modes and, control panel dialog box (Figure 4.14 (b)), is used to select operation modes and labeling options, perform camera settings such as zoom and rotate, perform 3D or 2D view selections and contains slider controls to move the three orthogonal planes during the centroid location mode. VCNS also provides several menu options for data storage and retrieval. The *File* menu provides sub-options for loading the segmented and non segmented volume data, and the look up table for names. The *Save Settings File* option under the *File* menu is used to save current state of the software. This creates a settings file, which contains information about the location of the current volume data file, segmented data file, current view settings and centroid location. The *Load Settings File* option helps in quickly restoring the previous state of the software

across multiple invocations. The *Help* menu opens up an online manual describing the usage of the system.



(a)



(b)

Figure 4.14. User Interface For VCNS

CHAPTER 5

SOFTWARE IMPLEMENTATION

This chapter describes the details of the design and implementation of VCNS. The system is developed in C++ and uses OpenGL application programming interface (API) for handling the graphics. A high level interaction diagram between various objects comprising the system is shown in the Figure 5.1. A more detailed diagram showing the relationship between the various classes for the whole system is shown in the appendix A. The following sections explain in detail each of the class objects used in the system.

5.1 Class Design

VCNS is designed using object oriented design (OOD) principles resulting in a flexible, and an easily maintainable system. The various classes defined by the system are explained in detail by following sections.

5.1.1 *PCIBirdInterface* Class

This class is a part of the tracker interface submodule of the tracking module and handles the task of interfacing with the pciBird^{TM} motion tracker system. It initializes the tracker with parameters such as the measurement rate, metric and hemisphere of operation during initialization. The measurement rate is fixed at 30 milliseconds to match with the display refresh rate of 30 frames per second. The distance metric is fixed as millimeters, while the hemisphere of operation is fixed as FRONT. The hemispheres arise because of the symmetry in the magnetic field around the transmitter. When the sensor is tracked continuously for its position, it is necessary to keep track of the transition of the sensor across the hemisphere boundary. This is done by monitoring the signs of X, Y and Z

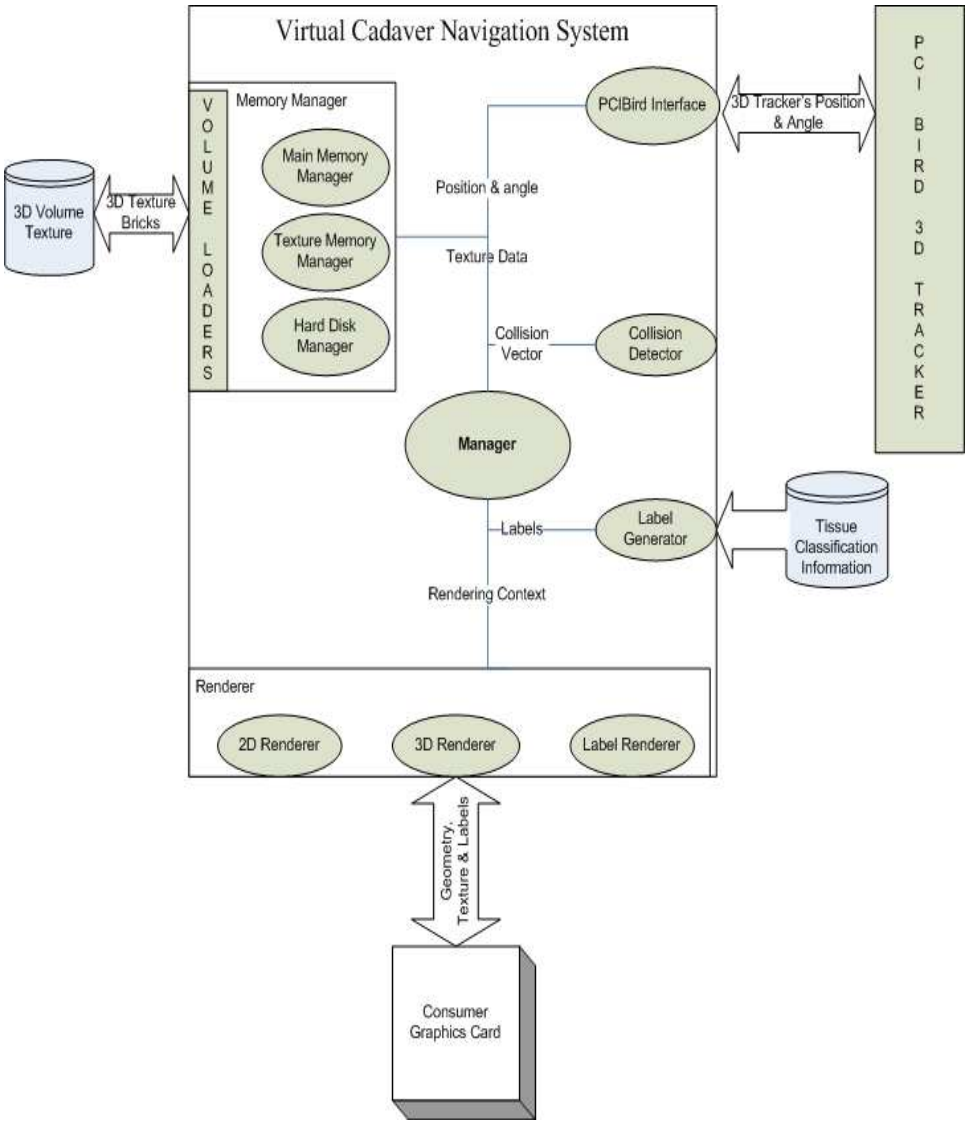


Figure 5.1. Interaction Diagram For Software Components Of VCNS

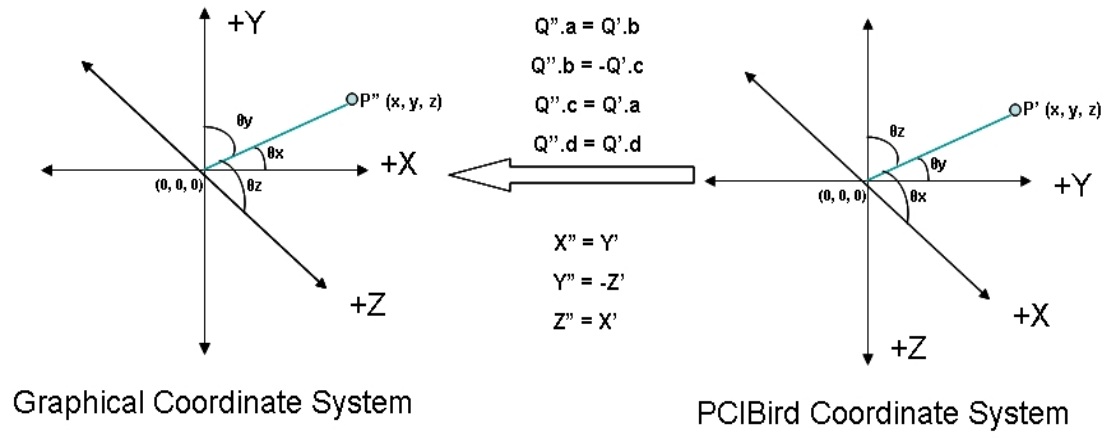


Figure 5.2. Conversion From PCIBird To OpenGL Coordinate System

coordinates of the tracker across successive measurements. When the boundary is crossed, the signs of both Y and Z measurements reverse, while that of X remains the same. The final values are obtained by reversing the signs of X, Y and Z. There is no change in the angles of the sensor when the transition occurs.

The *PCIBirdInterface* also performs necessary adjustments to synchronize the position of the tracker with its virtual counterpart. Figure 5.2 shows the conversions necessary to match the tracker coordinates with the OpenGL coordinates. Note that the angles are measured in quaternion notation to avoid "Gimbal Lock", which refers to a situation where it becomes impossible to rotate the object in a desired axis [5]. Euler method gives angle of rotation about the X, Y and Z axes separately, and are sensitive to the order in which the angles are applied. On the other hand, quaternion measures the rotation angle about an arbitrary axis of rotation and is free from the gimbal lock problem. The positions and the angles are queried by the *Manager* object during the tracking and labeling mode.

5.1.2 *MemoryManager* Class

This class and its subclasses are the part of resource management module. It is responsible for managing the resources such as: texture memory, graphics bus bandwidth, main memory and the hard disk. In order to perform this task, the functionality of the *Meme-*

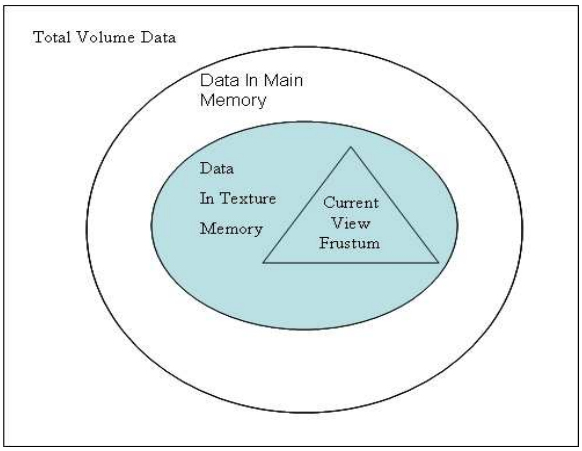


Figure 5.3. Hierarchial Distribution Of Volume Data In VCNS

oryManager class is divided into three classes: *HardDiskManager*, *MainMemoryManager* and *TextureMemoryManager* as shown in Figure 5.1.

When the volume data is loaded for the first time, the *HardDiskManager* divides it into bricks. The size of the bricks is decided on the factors such as size of texture memory, graphics bus speed and data bandwidth as well as the pixel fill rate of the graphics card. Large brick sizes require less seeks but are not interruptible, thereby slowing the visualization. Small brick sizes require more seeks but are more interruptible. The brick size of 64 X 64 X 64 gives a real time refresh rate of 30 fps.

Once the data is divided into bricks, it is written to a file called "brick file", which is more efficient in accessing the brick data than the ordinary volume data file. In addition, the *HardDiskManager* also maintains a mapping between the brick IDs and their offsets in the brick file. Bricks are assigned IDs/addresses by the *CollisionDetector* component during initialization. Using the brick file any brick can be accessed randomly using only one seek operation. The contagiously written bricks are also neighbors in the 3D space thereby providing a spatial coherence to certain extent. The *HardDiskManager* also augments the RGB data for the brick with the opacity values before writing, so that it can be displayed using OpenGL texture mapping routines. Black pixels (R = 0, G = 0, B = 0) are assigned an opacity of 0 (fully transparent) while the remaining pixels a value equal to 1 (completely opaque).

The *MainMemoryManager* component allocates the fixed amount of memory for the bricks from the process address space, and fragments it into slots of sizes equal to the brick size. These slots are maintained in the "least recently used" (LRU) list, and can contain only one brick at a time. Thus, there is a one-to-one mapping between the slot indices and the brick IDs. The LRU list is implemented using the heap data structure, with the top of the heap pointing to the least recently used slot with respect to a logical timer implemented as a counter variable. The logical timer is associated with each slot index and is updated every time it is accessed.

The *TextureMemoryManager* component queries the graphics hardware to find out the maximum available texture memory, and divides it into slots similar to the main memory slots. These slots are also arranged in a LRU list and have a one-to-one mapping to the brick IDs. At any instant, the LRU brick in the texture memory is the Most Recently Used (MRU) brick in the main memory. Thus, the bricks in the texture memory are a subset of the set of bricks present in the main memory as shown in the Figure 5.3. Note that, all the bricks in the texture memory may not be used depending upon the viewing frustum as shown in the Figure 5.3.

5.1.3 *VolumeDataLoader* Class

The VCNS offers flexibility of loading the volume data present in different image formats through the *VolumeDataLoader* class. In order to support a new file format, the custom data loader is implemented, by inheriting from the *VolumeDataLoader* and implementing the pure virtual function called *loadDataBrick*. The parameters to the *loadDataBrick* method accept the lower and upper limits of the sub volume of the data to be read from the file.

5.1.4 *CollisionDetector* Class

In addition to dividing the volume data into smaller data bricks, it is also necessary to divide the 3D geometry associated with it. Typically, the number bricks generated as a result of this division is large (1000-2000). While exploring the volume data using the

virtual cutting plane only a few bricks are intersected by the plane at a time. Hence, there is a need to organize the bricks more efficiently. The *CollisionDetector* component alleviates this problem by arranging the bricks in the form of a Binary Space Partition (BSP) tree (Figure 4.7). Each internal node in the tree represents a convex region in 3D space, which is subdivided into two child regions by means of a freely oriented partitioning plane [13]. The partitioning plane is one of the orthogonal planes (X or Y or Z) centered at the origin of the region. The root of the tree contains the complete volume geometry, while the leaf nodes contain the actual bricks. Each brick is also assigned a unique ID during the tree construction.

In order to determine the bricks intersected by the plane, the *CollisionDetector* recursively determines whether the plane intersects an internal node. This process, called collision/interference detection, is performed by the routines in the SOLID (SOftware Library for Interference Detection) library developed by [13]. If the region represented by the internal node is intersected by the cutting plane, its children are explored further; otherwise the entire subtree rooted at that internal node is ignored. Once the bricks intersected by the cutting plane are determined, it is clipped against the every brick using the Sutherland Hodgman 3D polygon clipping algorithm [56]. This results in a set of points representing the polygonal portion of the cutting plane inside that brick. The points of intersection are passed back to the *Manager* component during collision detection process.

5.1.5 *Renderer* Class

This class encapsulates the functionality of the visualization module. It exposes the rendering functionality through three child classes: *TMRenderer*, *CLMRenderer* and *LabelRenderer*. Every subclass implements the three functions called *beginDraw*, *draw* and *endDraw*. The *beginDraw* function is used to make the view settings such as camera position, rotation, scaling ,enabling 3D texturing etc that are specific to a renderer. The 3D texture mapping of the tomographic plane is implemented in the *draw* method, which takes the rendering context as an input parameter. The method *endDraw*, is mainly used to disable 3D texturing and undo the view settings done during the rendering.

The *TMRenderer* class is further divided into two subclasses named *TM2DRenderer* and *TM3DRenderer*, to draw the scene in 2D and 3D view respectively. In case of 2D renderer, the rotation and position of the virtual plane are negated, and multiplied with the "modelview" matrix, that is used in OpenGL for specifying the transformations [54]. Since the camera is fixed at origin, this transformation causes the plane to face the camera. This process is called as "billboarding" [1].

The *CLMRenderer* is used in the centroid locator phase, to display one or all of the three orthogonal planes through the volume. The *LabelRenderer* is a direct subclass of *TM2DRenderer* and extends its functionality by providing the functions to read back the texture mapped image and render label strings.

5.1.6 *LabelGenerator* Class

The *LabelGenerator* component generates the labels from the segmented slice by identifying the groups of pixels belonging to the same anatomical part. To enable this component, it is necessary to load the lookup table for mapping color values with the names. The *LabelGenerator* reads the mapping file and creates a map for storing the name and color associations. It returns a set of labels for the various tissues in the slice during the labeling mode. These labels are superimposed on the non segmented data slice by the *LabelRenderer* object.

5.1.7 *Manager* Class

This component is responsible for coordinating all the activity in the system and encapsulates the functionality of the tracking, centroid locator and the labeling modules. When the non segmented and the segmented data are loaded the manager creates the instances of *MemoryManager*, *CollisionDetector* and the *Renderers* class. During the rendering, the manager creates a "rendering context" containing the bricks currently intersected by the tomographic plane, the points of intersection of the plane with every brick and the OpenGL texture objects, that point to the area of texture memory representing the texture data for the brick. This context is passed to a suitable renderer depending upon the current

mode of operation. The role of *Manager* component is discussed in detail, in the context of software implementation.

5.2 Software Implementation

This section presents the low level details of implementation for each of the three operating modes.

5.2.1 Tracking Mode

In this mode, the pciBird is initialized and probed periodically to obtain the position and angles of the sensor. The following steps are performed in order to map virtual plane emanating from the probe, with the texture data from the bricks.

Step 1. The *Manager* reads pciBird's position and orientation, and computes the plane geometry based on this data.

Step 2. The *Manager* queries the *CollisionDetector*, to find out the bricks intersecting with the plane, and stores them in collision vector. The *CollisionDetector* obtains this information by performing a collision detection query with every internal node of the BSP tree until it reaches the leaf node containing the desired brick (Figure 4.8). The intersecting bricks are returned along with the points of intersection as a part of *CollisionData* data structure.

Step 3. The *Manager* creates batches of bricks, each of size equal to maximum number of bricks that can be accommodated in the texture memory. Every batch is passed to the *MemoryManager* for binding the texture data. The *MemoryManager* first arranges the bricks such that the bricks present in the texture memory are at the front of the batch, followed by those present in the main memory, and finally the bricks on the hard disk. After prioritizing the processing order of bricks, texture binding is performed for those not in the texture memory. The brick is located in the main memory and/or the hard disk, and the texture data is loaded into the next available texture memory slot by the *TextureMemoryManager*.

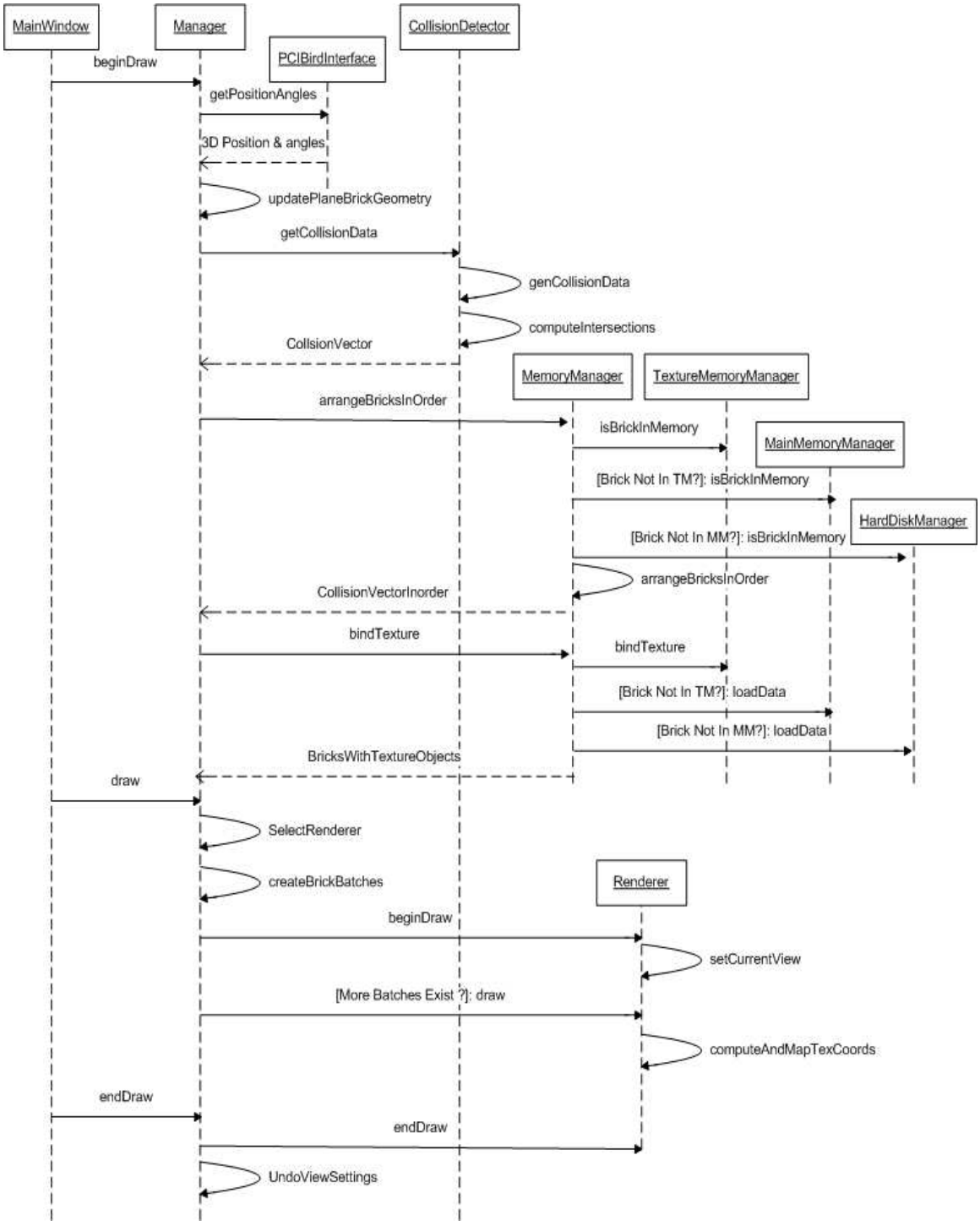


Figure 5.4. Sequence Diagram For Tracking Mode Based On UML Principles. An Underline Below A Class Name Implies That The Instance Of That Class Is Used

Step 4. The *Manager* passes the rendering context to one of the 2D or 3D tracking mode renderers depending upon the current view settings. The renderer normalizes the points of intersections between 0 and 1 as per the requirements of OpenGL. This is done by subtracting position of the center of the brick and scaling it down by the dimensions of the brick. The 3D texture mapping is performed by specifying every point of intersection of the plane and brick, and its corresponding texture coordinate.

Figure 5.4 shows the messages passed between various objects during the process of tracking.

5.2.2 Centroid Location Mode

The centroid location mode is used to align the origin of volume data with that of the mannequin. When the volume data is loaded, the origin of the volume is at the center of the box enclosing the volume, which may not be the location of center of the pciBird transmitter and mannequin. The centroid location mode computes the slices along the transaxial (XY), sagittal (YZ) and coronal (XZ) planes. These planes can be moved using slider controls in the control panel dialog box (Figure 4.14 (b)), to match the two coordinate systems. The point where the three planes meet, is considered as the origin of the volume data during tracking mode. Corrections are made to the position obtained from the pciBird based on the location of centroid of volume data. The messages passed between various objects is exactly same as the tracking mode shown in the Figure 5.4, except that the positions of the planes are queried from the control dialog box instead of the tracker.

5.2.3 Labeling Mode

In this mode, it is necessary to load the segmented volume data and the mapping between the segmented values and the names. The segmented data is also divided into bricks similar to the non segmented data and written to the disk in a separate brick file. The non segmented data bricks, however, share the geometry with the segmented data bricks. Thus, it is necessary to load the non segmented data before loading the segmented

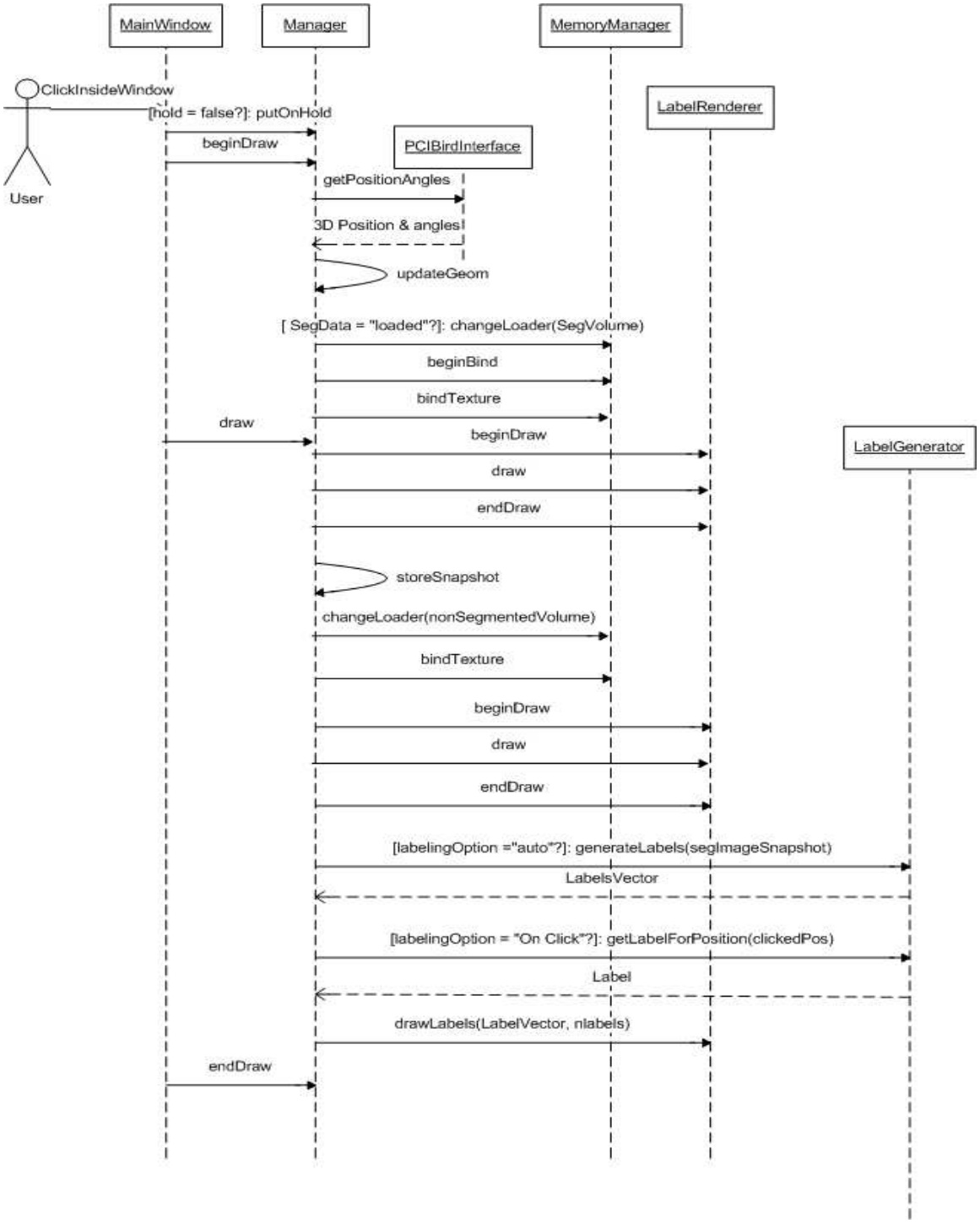


Figure 5.5. Sequence Diagram For Labeling Mode Based On UML Principles. An Underline Below A Class Name Implies That The Instance Of That Class Is Used

data. The software ensures the sequence by disabling the *Load Segmented Volume* menu option, until the non segmented data is loaded using the *Load Volume* menu option.

In order to generate the labels, it is necessary to freeze the motion of the probe by clicking inside the window with the left mouse button. This puts the manager on the hold. As a result, the manager captures the snapshot of the 2-D slice through the segmented dataset at the frozen position and orientation of the probe. The colored slice is obtained from the non segmented dataset in the same way as the tracking mode.

Depending upon the current labeling option, either all the anatomical parts on the slice are labelled, or only those selected by the user using the right mouse button. In the case of automatic labeling, all the anatomical parts on the current slice are identified from the segmented slice and labelled by a look up operation in the table containing mapping between the name and RGB values. This mapping is created from the look up table file loaded from the *Load LUT* menu option. In case of "On-Click" labeling option, the label is generated only for the selected structure. The sequence diagram (Figure 5.5) shows the messages exchanged between various objects during the labeling process.

CHAPTER 6

RESULTS AND DISCUSSION

This chapter presents the outputs and the performance analysis for various modes of operation of VCNS. The advantages of using VCNS as a teaching tool, and the possible amendments to it are discussed in the end.

6.1 Results

The tomographic slices resulting from the interaction of the probe with the volume data for the visible male, in the tracking mode, are shown in the Figure 6.1, Figure 6.2. The image in Figure 6.1 shows the cross sectional view of the lungs, and Figure 6.2 shows the cross sectional view of the vertebral column. These figures depict how the probe can be used to obtain the cross sectional slices at various angles and positions. Figure 6.3 shows how the label generator mode can be used to identify the lower lobe of the left lung. Figures 6.5 through Figure 6.7 show the transaxial, sagittal and coronal slices through the volume, which are used to align the centroid of the volume data during the centroid locator process. The point of intersection of the three orthogonal planes is used to determine the position of the centroid as shown in the Figure 6.4. This position is where the transmitter of the tracker system is placed.

6.2 Performance Analysis

The factors affecting the performance of VCNS are as follows:

1. Size of the brick
2. Size of the texture memory

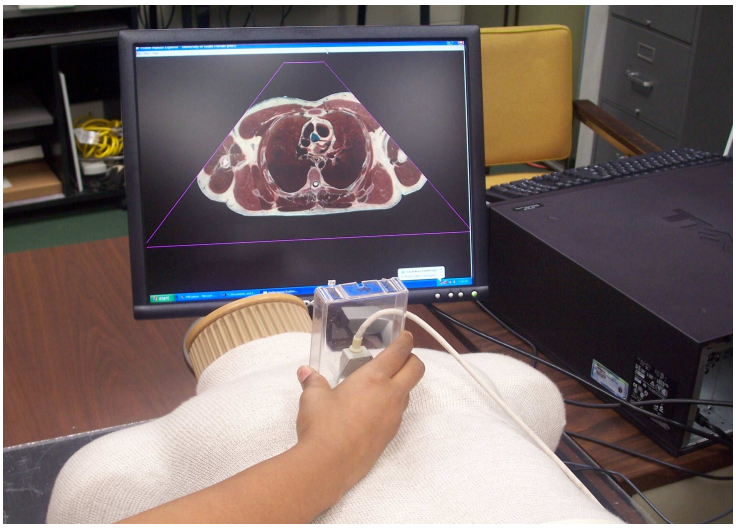


Figure 6.1. A Tomographic Slice Of Lung In Tracking Mode Through Visible Male Data

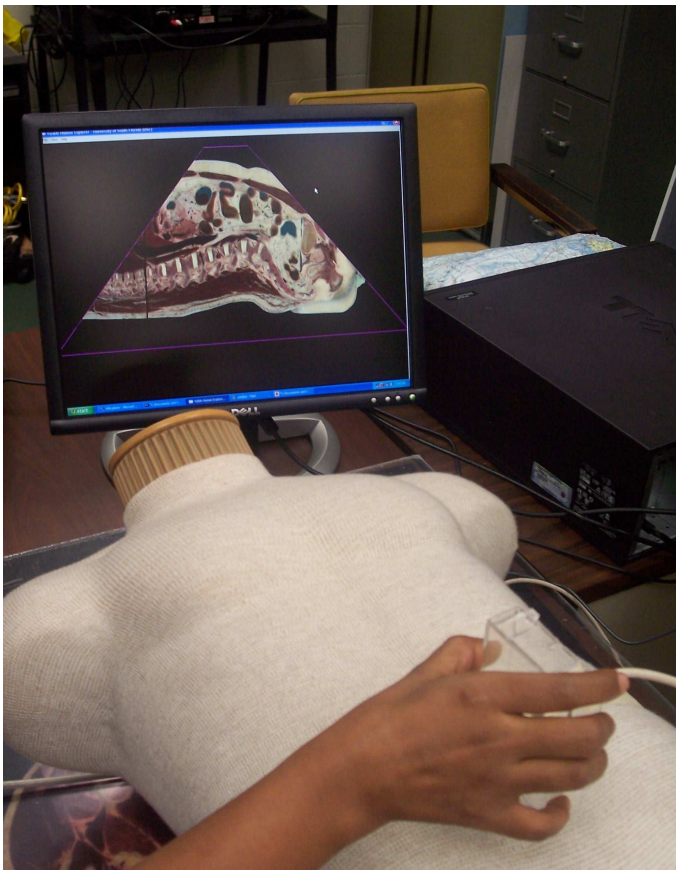


Figure 6.2. A Tomographic Slice Of Vertebral Column In Tracking Mode Through Visible Male Data

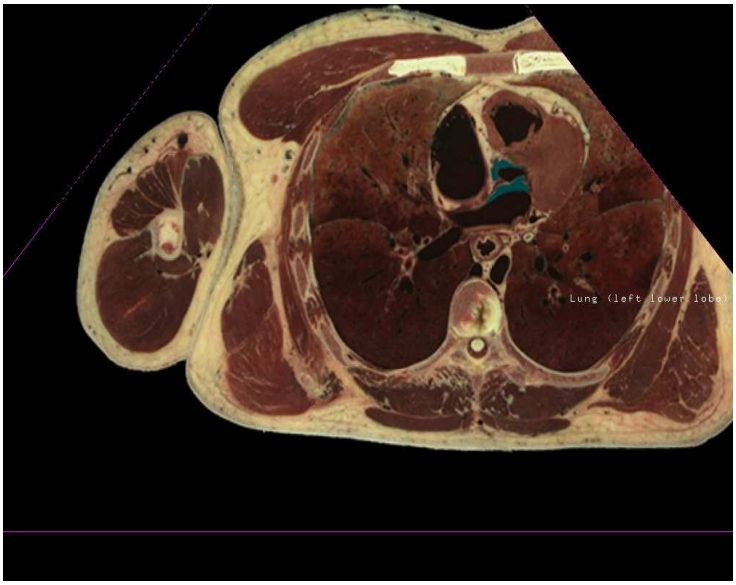


Figure 6.3. Identification Of Lower Left Lobe Of Lung Using The Labeling Tool

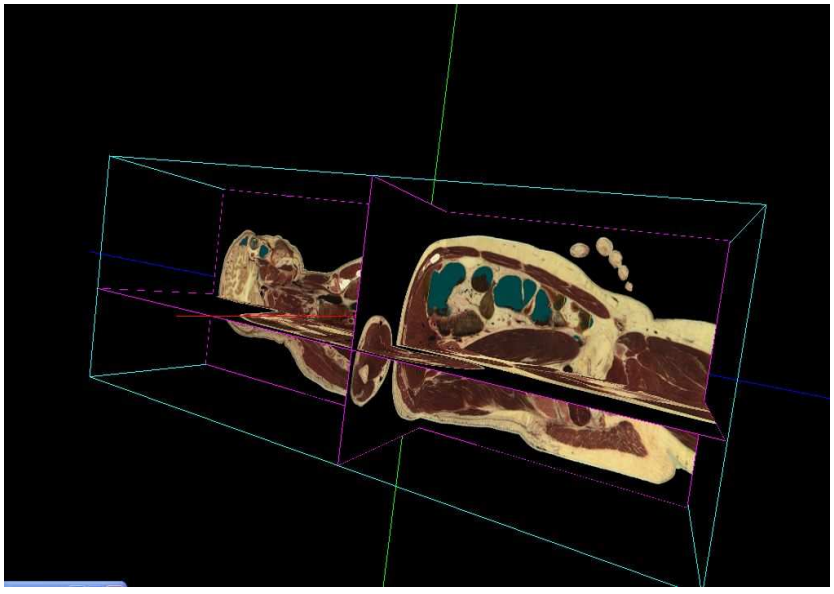


Figure 6.4. Diagram Showing The Intersection Of The Three Orthogonal Planes Through The Visible Male During Centroid Locator Mode

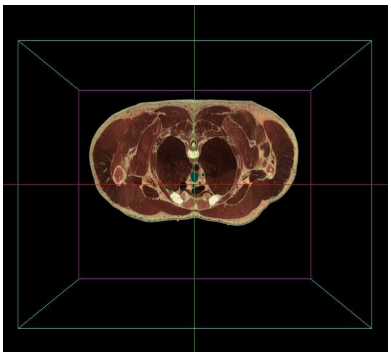


Figure 6.5. Transaxial Slice Through The Visible Male Data Generated During Centroid Locator Mode

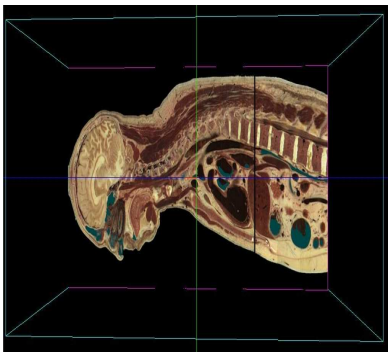


Figure 6.6. Sagittal Slice Through The Visible Male Data Generated During Centroid Locator Mode

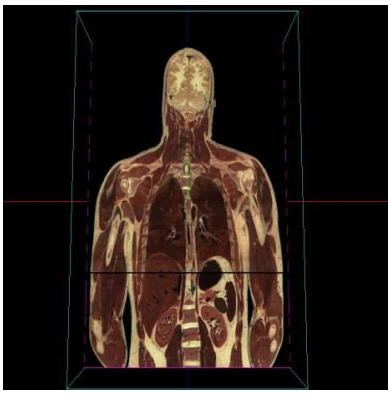


Figure 6.7. Coronal Slice Through The Visible Male Data Generated During Centroid Locator Mode

3. Size of the main memory

4. Bandwidth of the graphics bus

The criteria for performance is considered as frame rate, which is measured as the number of frames that can be rendered per second. An average time for rendering a frame is calculated by averaging the difference between the system time before and after rendering a frame over 1000 frames. The frame rate is then calculated as the reciprocal of the average time for rendering a frame. The frame rate is measured for each of the operating modes separately by varying the various parameters mentioned above. The performance analysis is performed for two test machines with different configurations as shown in the Table 6.1.

Table 6.1. Configurations Of Two Test Machines Used For Performance Analysis

Configuration	Machine 1	Machine 2
Processor	Intel Pentium 1.5GHz	Intel Pentium Dual Processor 3.6GHz
Main Memory	1GB	4GB
Graphics Card	ATI Radeon 8700, 64MB, AGP 2X	NVIDIA Quadro FX 4400, 512 MB, PCI Express 16X
Storage	Seagate, SCSI, 7200 RPM, 160GB	Seagate Ultra SCSI, 15000 RPM, 73GB

Figures 6.10, 6.11 show the impact of brick size on frame rate for both the test boards. Figure 6.11 indicates that for Radeon 8700 an increase in the size of the brick till 64 X 64 X 64 increases the frame rate. However, beyond this size the frame rate decreases to

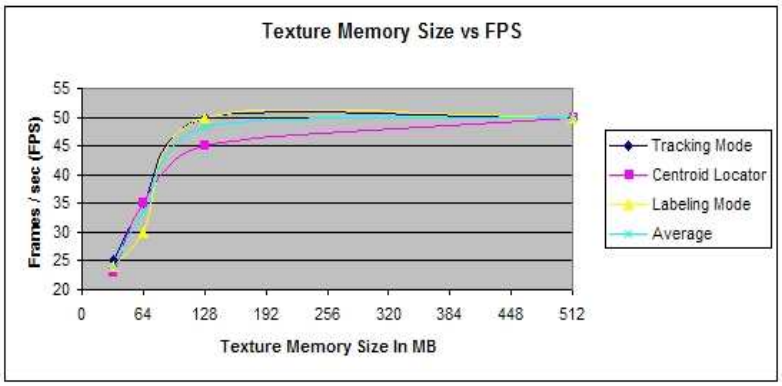


Figure 6.8. Variation Of Frame Rate With Texture Memory Size On NVIDIA Quadro FX 4400 (Main Memory = 512MB, Brick Size = 64 X 64 X 64)

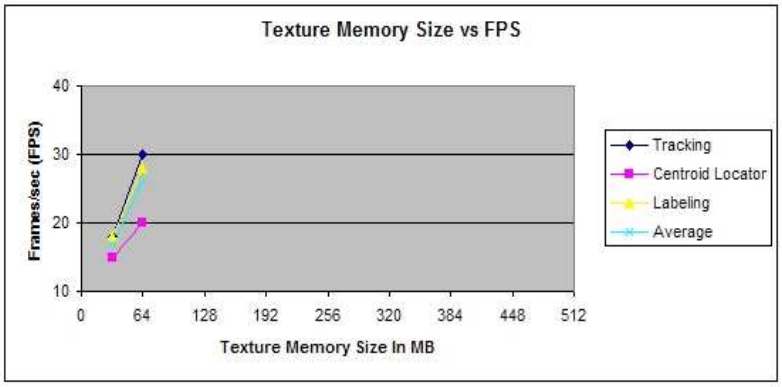


Figure 6.9. Variation Of Frame Rate With Texture Memory Size On ATI Radeon 8700 (Main Memory = 512MB, Brick Size = 64 X 64 X 64)

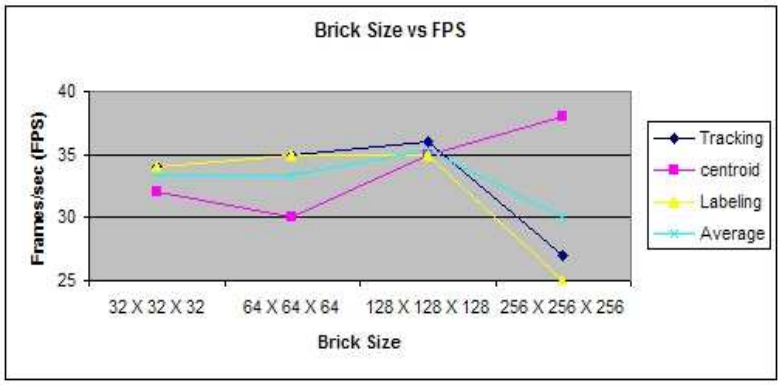


Figure 6.10. Variation Of Frame Rate With Brick Size On NVIDIA Quadro FX 4400 (Main Memory = 512MB, Texture Memory = 64MB)

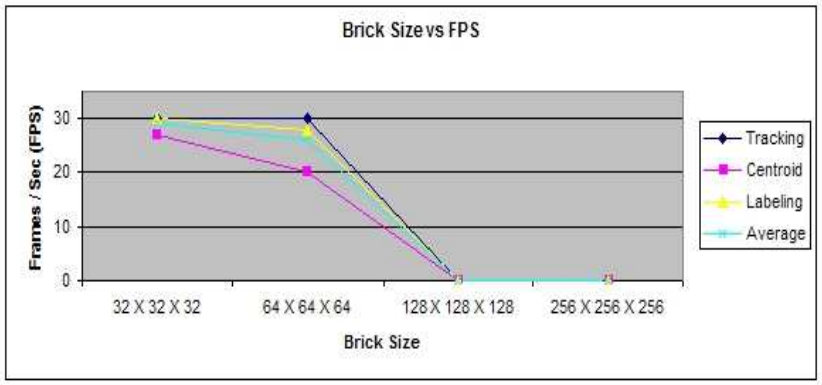


Figure 6.11. Variation Of Frame Rate With Brick Size On ATI Radeon 8700 (Main Memory = 512MB, Texture Memory = 64MB)

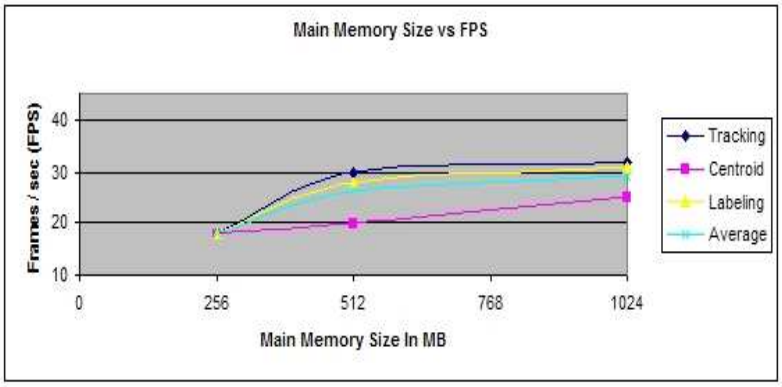


Figure 6.12. Variation of Frame Rate With Main Memory Size On NVIDIA Quadro FX 4400 (Texture Memory = 64MB, Brick Size = 64 X 64 X 64)

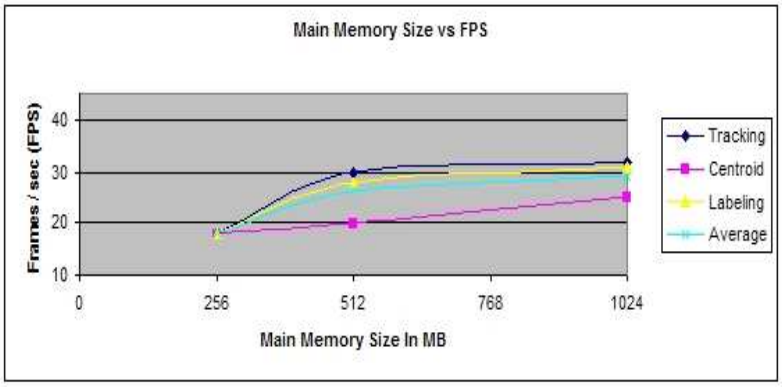


Figure 6.13. Variation Of Frame Rate With Main Memory Size On ATI Radeon 8700 (Texture Memory = 64MB, Brick Size = 64 X 64 X 64)

zero, which is due to the limitations offered by the graphics bus bandwidth. In case of the NVIDIA Quadro, this limit is higher because of the higher bandwidth offered by the PCI Express 16X graphics bus as shown in the Figure 6.10. The effect of size of texture memory on the frame rate is shown in the Figures 6.8, 6.9. The increase in the texture memory increases the frame rate substantially (almost by 10 frames per second) in case of both the graphics cards. Figures 6.12, 6.13 shows the impact of main memory size on the frame rate. An increase in the size of the main memory from 512 MB to 1 GB, only, minimally increases the frame rate in both the cases.

All of the above figures indicate that the frame rate is lower in case of the centroid locator than the tracking and labeling mode in most cases. In the Figure 6.10, however, for brick size of 256 X 256 X 256 the frame rate is higher in case of the centroid locator mode than the other modes, since the bricks are accessed less randomly, thereby decreasing the frequency of the downloads for the bricks. The frame rate in labeling mode is always in par with the tracking mode, although the segmentation dataset is also used along with the non segmented data.

The performance analysis shows that VCNS is able to achieve interactive frame rates (30 fps or more) on both the test machines, with atleast 64MB of texture memory and 512MB of main memory.

6.3 Discussion

The proposed VCNS system is a cost effective system capable of exploring full resolution visible male and visible female cadaveric data on a personal computer (PC). The ability to point at the mannequin and obtain tomographic slice at arbitrary locations and orientation helps in understanding spatial locations of various organs, which is essential for learning the human anatomy. The integration of the segmentation data helps in obtaining labels, making the learning process accessible even for a novice. The system can be easily extended to support different file formats and can be used to explore clinical data of patients for diagnostic purposes such as CT and MRI. The software can also be used to teach ultrasound

imaging technique for students and technicians. The system is flexible, portable and can be installed easily on any workstation with at least 64 Megabytes of texture memory and 512 Megabytes of main memory.

The system can be extended to provide more information to improve the understanding. For example, supplementary information such as histology images, brief descriptions of functions of organs on the slice can be linked with the slice. 3D models of organs and bones can also be integrated to provide a more intuitive interface. The system can be easily extended to include an evaluation tool that can grade the student, based his/her ability to locate, and label the organs correctly by pointing at the mannequin.

REFERENCES

[1] http://www.3dlabs.com/tutorials/billboarding.htm.

[2] http://www.ascension.com.

[3] http://www.libtiff.org/.

[4] http://www.m-w.com/.

[5] http://www.mathworld.com.

[6] http://www.nlm.nih.gov.

[7] M. J. Ackerman. The visible human project. *Journal of Biocommunication*, 18(2):14, 1991.

[8] Adobe Systems Incorporated. *TIFF Revision 6.0*, June 1992.

[9] M. Alcaniz, C. Perpina, R. Banos, J. A. Lozano, J. Montesa, and e. a. Botella, C. A new realistic 3d body representation in virtual environments for the treatment of disturbed body image in eating disorders. *CyberPsychology and Behavior*, 3(3):421–432, 2000.

[10] M. R. Ali, Y. Mowery, B. Kaplan, and E. J. DeMarial. Training the novice in laparoscopy. *Surgical Endoscopy*, 16(12):1732–1736, December 2002.

[11] Ascension Technology. *PCI Bird Technical Reference*.

[12] R. Bargar, I. Choi, S. Das, and G. Camille. Model-based interactive sound for an immersive virtual environment. *Proceedings of the 94 International Computer Music Conference*, 1994.

[13] G. V. D. Bergen. *Collision Detection in Interactive 3D Environments*. Morgan Kaufmann, October 2003.

[14] C. Botella, C. Perpina, R. M. Banos, and A. Garcia-Palacios. Virtual reality: a new clinical setting lab. *Studies in Health Technology and Informatics*, 58:73–81, 1998.

[15] C. Chinnock. Virtual reality in surgery and medicine. *Hospital Technology Series*, 13(18):1–48, 1994.

[16] S. L. Delp and F. R. Zajac. Force and moment generating capacity of lower limb muscles before and after tendon lengthening. *Clinical Orthopaedic and Related Research*, 284:247–259, 1992.

[17] A. M. DiGioia, B. Jaramaz, R. V. O'Toole, D. A. Simon, and T. Kanade. Medical robotics and computer assisted surgery in orthopaedics: an integrated approach. *In: Interactive Technology and the New Paradigm for Healthcare. IOS Press. Washington D. C.*, pages 88–90, 1995.

[18] H. D. Dobson, R. K. Pearl, C. P. Orsay, M. Rasmussen, R. Evenhouse, and Z. Ali. Virtual reality: a new method of teaching anorectal and pelvic floor anatomy. *Dis Colon Rectum*, 46(3):349–352, 2003.

[19] B. J. Dunkin. Flexible endoscopy simulators. *Semin Laparosc Surg*, 10(1):29–35, 2003.

[20] P. M. Emmelkamp, M. Bruynzeel, L. Drost, and C. A. Van der Mast. Virtual reality treatment in acrophobia: a comparison with explore in vivo. *CyberPsychology and Behavior*, 4(3):335–339, 2001.

[21] A. Garcia-Palacios, H. Hoffman, A. Carlin, T. A. Furness, and C. Botella. Virtual reality in treatment of spider phobia: a controlled study. *Bhavior research and Therapy*, 40(9):983–993, 2002.

[22] A. V. Gelder and K. Kim. Direct volume rendering with shading via three-dimensional textures. In *VVS '96: Proceedings of the 1996 symposium on Volume visualization*, pages 23–ff., Piscataway, NJ, USA, 1996. IEEE Press.

[23] J. C. Goble, K. Hinckley, R. Pausch, J. W. Snell, and N. F. Kassell. Two-handed spatial interface tools for neurosurgical planning. *IEEE Computer*, 28(7):20–26, July 1995.

[24] M. Gor, R. McCloy, R. Stone, and A. Smith. Virtual reality laparoscopic simulator for assessment in gynaecology. *BJOG: An International Journal Of Obstetrics And Gynaecology,*, 110(2):181–187, February 2003.

[25] D. Gross. *Technology management and user acceptance of VE technology.*, chapter Handbook of Virtual Environments: Design, Implementation, and Applications. Lawrence Erlbaum Associates, Inc., Mahwah, N.J., 2002.

[26] S. Halligan and H. M. Fenlon. Virtual colonoscopy. *British Medical Journal*, 319(7219):1249–1252, 1999.

[27] K. H. Hohne. *VOXEL-MAN, Part 1: Brain and Skull Version 1.1 .* Springer-Verlag Electronic Media, 1996.

[28] P. R. Jeffries, S. Woolf, and B. Linde. Technology based vs traditional instruction. a comparison of two methods for teaching the skill of performing a 12-lead ecg. *Nursing Education Perpective*, 24(2):70–74, 2003.

[29] N. W. John, N. Thacker, M. Pokric, A. Jackson, G. Zannetti, and E. e. a. Gobbetti. An integrated simulator for surgery of the petrous bone. *Stud Health Technol Inform*, 81:218–224, 2001.

[30] Krueger. Olfactory stimuli in virtual reality medical training. Report, ARPA, 1994.

[31] P. Lacroute and M. Levoy. Fast volume rendering using a shear-warp factorization of the viewing transformation. In *SIGGRAPH '94: Proceedings of the 21st annual conference on Computer graphics and interactive techniques*, pages 451–458, New York, NY, USA, 1994. ACM Press.

[32] E. LaMar, M. A. Duchaineau, B. Hamann, and K. I. Joy. Multiresolution techniques for interactive texture-based rendering of arbitrarily oriented cutting planes. *Proceedings of VisSym*, July 1998.

[33] C. H. Lewis and M. J. Griffin. *Human factors consideration in clinical applications of virtual reality*, chapter Virtual reality in neuro-psycho-physiology., pages 35–56. Amsterdam:IOS Press, 1997.

[34] W. E. Lorensen. Marching cubes: A high resolution 3d surface reconstruction algorithm. *SIGGRAPH*, 21(4):163–169, 1987.

[35] W. E. Lorensen, F. A. Jolesz, and R. Kikinis. The exploration of cross-sectional data with a virtual endoscope. *In: Interactive Technology and the New Paradigm for Healthcare*, pages 221–230, 1995.

[36] N. Maltby, I. Kirsch, M. Mayers, and G. Allen. Virtual reality exposure therapy for the treatment of fear of flying: A controlled investigation. *Journal of Consulting and Clinical Psychology*, 70(5):1112–1118, 2002.

[37] J. Moline. Virtual reality for health care: a survey. *Virtual Reality in Neuro-Psycho-Physiology. Amsterdam:IOS Press;*, pages 3–37, 1998.

[38] K. Moorthy, S. Smith, T. Brown, S. Bann, and A. Darzi. Evaluation of virtual reality bronchoscopy as a learning and assessment tool. *Respiration*, 70(2):195–199, 2003.

[39] H. Nicolas, S. ad Patel. Health and safety implications of virtual reality : a review of empirical evidence. *Applied Ergonomics*, 33(3):251–271, 2002.

[40] J. C. Norcross, M. Hedges, and J. O. Prochaska. The face of 2010: A delphi poll on the future of psychotherapy. *Professional Psychology: Research and Practice*, 33(3):316–322, 2002.

[41] L. Piron, F. Ceni, P. Tonin, and M. Dam. Virtual reality as an assessment tool for arm motor deficits after brain lesions. *Studies in Health Technology and Informatics*, 81:386–392, 2001.

[42] G. Riva. Virtual environments in neuroscience. *IEEE Transactions on Information Technology in Biomedicine*, 2(4):275–281, 19989.

[43] G. Riva. Applications of virtual environments in medicine. *Methods Of Information In Medicine*, 42(5):524–534, 2003.

[44] G. Riva, M. Bacchetta, M. Baruffi, and E. Molinari. Virtual-reality-based multidimensional therapy for the treatment of body image disturbances in binge eating disorders: a preliminary controlled study. *IEEE Transactions on Information Technology in Biomedicine*, 6(3):224–234, 2002.

[45] A. A. Rizzo and J. G. Buckwalter. *Virtual reality and cognitive assessment and rehabilitation: the state of the art.*, chapter Virtual reality in neuro-psycho-physiology., pages 123–146. Amsterdam:IOS Press, 1997. Online: http://www.cybertherapy.info/pages/book1.htm.

[46] M. D. Ross, I. A. Twombly, C. Bruyns, R. Cheng, and S. Senger.

[47] B. O. Rothbaum, L. Hodges, S. Simth, J. H. Lee, and L. Price. A controlled study of virtual reality exposure therapy for the fear of flying. *Journal of Consulting and Clinical Psychology*, 68(6):1020–1026, 2000.

[48] F. Rubino, L. Soler, J. Marescaux, and H. Maisonneuve. Advances in virtual reality are wide ranging. *British Medical Journal*, 324(7337):612, 2002.

[49] R. Satava and S. Jones. Current and future applications of virtual reality for medcine. *Proceedings of the IEEE*, 86(3):484–489, march 1998.

[50] R. M. Satava. Virtual reality surgical simulator. *Surgical Endoscopy*, 7:203–205, 1993.

[51] R. M. Satava and S. B. Jones. Medical applications of virtual reality. In K. M. Stanney, editor, *Design, Implementation, and Application*, pages 368–391. Lawrence Erlbaum Associates Inc, 2002.

[52] M. T. Schultheis and A. A. Rizzo. The application of virtual reality technology in rehabilitation. *Rehabilitation Psychology.*, 46(3):296–311, 2001.

[53] W. R. Sherman and A. B. Craig. *Understanding Virtual Reality : Interface, Application, And Design*. Morgan Kaufmann, 2003.

[54] D. Shreiner, M. Woo, N. Jackie, and T. Davis. *OpenGL Programming Guide: The Official Guide to Learning OpenGL, Version 1.4*. Addison-Wesley Professional, 4th edition, 2003.

[55] V. Spitzer, M. J. Ackerman, A. L. Scherzinger, and D. Whitlock. The visible human male: a technical report. *Journal of American Medical Informatics Association*, 3(2):118–130, 1996.

[56] I. E. Sutherland and G. W. Hodgman. Reentrant polygon clipping. *Commun. ACM*, 17(1):32–42, 1974.

[57] M. Teistler, O. J. Bott, J. Dormeier, and D. P. Pretschner. Virtual tomography: A new approach to efficient human-computer interaction for medical imaging. *Proceedings of SPIE*, 5029:512–519, 2003.

[58] M. Teistler and D. P. Pretschner. Visualization of echocardiographic data using virtual scenes. *Computers In Cardiology*, 26:399–402, 1999.

[59] J. Vince. *Virtual Reality Systems*. SIGGRAPH Series. Addison-Wesley, 1st edition, 1995.

[60] F. Vincelli, L. Anolli, S. Bouchard, B. K. Wiederhold, V. Zurloni, and G. Riva. Experimental cognitive therapy in the treatment of panic disorders with agrophobia: A controlled study. *CyberPsychology and Behavior*, 6(3):312–318, 2003.

[61] F. Vincelli, E. Molinari, and G. Riva. Virtual reality as clinical tool: immersion and three-dimensionality in the relationship between patient and therapist. *Studies in Health Technology and Informatics*, 81:551–553, 2001.

[62] W. R. Volz. Gigabyte volume viewing using split software/hardware interpolation. In *VVS '00: Proceedings of the 2000 IEEE symposium on Volume visualization*, pages 15–22, New York, NY, USA, 2000. ACM Press.

[63] B. K. Wiederhold, D. P. Jang, S. I. Kim, and M. D. Wiederhold. Physiological monitoring as an objective tool in virtual reality therapy. *CyberPsychology and Behavior*, 5(1):77–82, 2002.

[64] R. Wootton. *European Telemedicine*, chapter Telemedicine: an introduction, pages 10–12. Kensington Publications Ltd., London, 1998/99.

[65] L. Zhang, B. C. Abreu, B. Masel, R. S. Scheibel, C. H. Christiansen, and H. N. Virtual reality in assessment of selected cognitive function after brain injury. *American Journal of Physical Medicine and Rehabilitation*, 80(8):597–604, 2001.

APPENDICES

Appendix A Class Diagram for VCNS

The class diagram for the system based on unified modeling language design principles is shown in the Figure A

Appendix A (Continued)

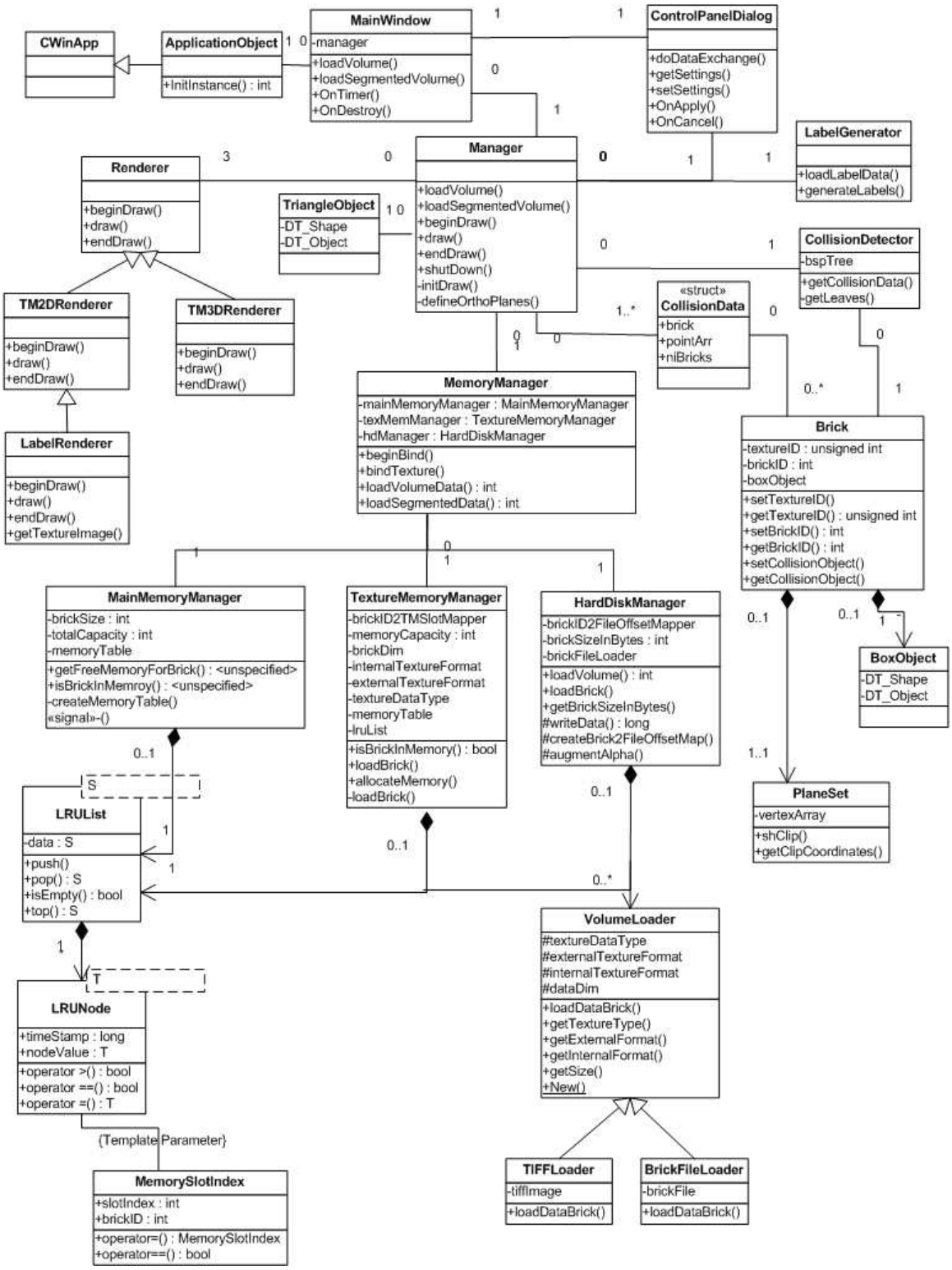


Figure A.1. VCNS Class Diagram