Computer Visualization Techniques in Surgical Planning for Pedicle Screw Insertion

Choi Yi King

A dissertation submitted to
the University of Hong Kong
in partial fulfillment of the requirements for
the degree of Master of Philosophy

August 2000

Abstract of thesis entitled
"Computer Visualization Techniques
in Surgical Planning for Pedicle Screw Insertion"

submitted by Choi Yi King

for the degree of Master of Philosophy
at The University of Hong Kong in August 2000

To many people, computer graphics may be merely the basis for compositing visual entertainment. TV ads, movies, and computer games are some of these examples. In fact, computer graphics techniques can also be observed from many other real-life applications such as astronomical studies, and architectural and mechanical designs. Medical visualization is one of these applications for which many of start-of-the-art computer graphics techniques are developed.

Medical visualization involves the use of computer graphics techniques to enhance the interpretation and manipulation of the vast amount of medical data acquired by various imaging techniques. Many systems specializing in medical visualization have been built. While general-purpose commercial visualization software can provide common functions for visualizing three-dimensional (3D) medical data, there are many occasions where specific requirements cannot be fulfilled by such software. There are many different aspects of studies within the medical discipline, each focusing on a particular structure of the human body. And for a single subject — for example, the eye — more than one properties may be emphasized. For instance, in orthopaedic surgery involving screw insertion, surgeons must consider not only the surface topology but also the bone mineral density (BMD) distribution to ensure a high success rate of the operation. However, this vital information is not observed in commercially available products for assisting surgeons in the decision-making process of the surgical operations.

In view of this, a joint research project is conducted with medical specialists of the Department of Orthopaedic Surgery at HKU. The aim of the project is to develop a customized computer visualization system, called *VISBONE*, to enhance surgical operation planning for pedicle screw insertion.

In this thesis, an experimental system has been built for the purpose as described. Design issues and specific considerations for this special application are discussed. Apart from applying computer visualization techniques, we also introduce new methods to tackle some specific problems and difficulties that have arisen. This work finds its significance in linking research in orthopaedics with that in computer graphics. It proves the feasibility of applying computer visualization techniques to orthopaedic surgical planning. It also leads to a call for turning the experimental system into a clinical application. The clinical significance of the system will then be evaluated. We hope that the patients of related surgeries can benefit from these studies.

To my family

Contents

Chapter 1

Introduction

Technology advancements have continuously changed our mode of livings. Among these, the introduction of microprocessor chips has the most direct impacts on our everyday lives. Computers brought to us unimaginable processing powers that are not only important to our daily lives but even to the evolution of the entire human race (e.g. DNA-code breaking). Its possible applications in every aspects are being explored. Imagine the times when there is no computer. People often have to rely only on their experience and expertise to accomplish their tasks. For instance, villagers and oil companies had to drill holes randomly in the hope of bumping into underground water or oil by chance. This was the same in the medical disciplines where surgeons had to open up the body of a patient before knowing what was happening inside. Life was much easier when medical imaging came into place. Imaging techniques like X-ray, Magnetic Resonance Imaging (MRI) and Computed Tomography (CT) all provide surgeons with an "underground map" for pre-surgical comprehension. However, these 2D static images are still not enough. During a surgical operation, what surgeons are facing are those 3D vivid organs. It is not an easy task to relate 2D images with 3D structures. Spatial orientations and relationships that are important to surgeons are lacking in these 2D images.

Computer assisted surgery (CAS) consists of three stages: pre-operative planning, intra-operative guidance and post-operative evaluations. It incorporates different techniques which can be classified into five components: visualization, localization, access, control, and integration [1]. Much research is conducted to integrate the components for providing information-rich and comprehensive assistance to all related tasks of the three surgical stages. Among the five components, one of the major objectives for visualization is to fill up the gap between 2D images acquired by various medical imaging techniques and the required 3D sensations as mentioned above.

Throughout the past years, the applications of computer assisted surgery have penetrated various medical disciplines, e.g. eye surgery, dentistry, neurosurgery, and many others. Although the basic principles for applying CAS are more or less the same, there are in fact different specific requirements for different kinds of surgery. In brain surgeries, for instance, the computer system must be able

to differentiate between white matter and grey matter. In cardiac operations, surgeons are interested in knowing the blood-bumping activities performed by the heart.

In orthopaedic surgeries involving bone injuries, some focus on the bone surface topology while the others may concern about internal bone tissues. For total bone replacement, like pelvic bone replacement, surgeons would like to know the shape of the infected or injured bone so that they can reshape and polish an artificial bone for replacement which fits the original. Many commercial products[1] possess sophisticated features for these tasks. These software products all provide surface-based models to show the shape of a bone. However, in other orthopaedic surgeries involving screw insertions, information on internal bone tissues is more important than the surface topology. This piece of information cannot be shown simply by a surface-based visualization.

A special operation in spine or pelvic bone surgery is to insert screws into the injured bones to maintain skeletal structures. In case of scoliosis, a patience's spine has abnormal curves with a rotational deformity. He therefore suffers from lack of abilities to support his own bodies. Scoliosis is commonly treated by pedicle screw insertion. The surgery is performed using a posterior approach meaning that the patient's body is opened from the back and the back part of the vertebrae is exposed. Surgeons then drill holes and insert the screws into the pedicle in a proper direction (Figure 1.1). They do this by using their anatomic knowledge

Figure 1.1: The top and side views of a vertebrae. A screw has been inserted to the pedicle.

and must be very careful as the spinal cord must not be damaged during the process. These screws are then used to hold other implantable instruments like rods and hooks for straightening out the spine. Without any intra-operative visual assistance, this kind of insertions is highly prone to errors. This is because minor changes in screw orientation will lead to significant deviations for the screw tip. Moreover, there are many variations that the surgeons must consider for the operations. The width, height and spatial orientation of the pedicle differ

[1] An example is the Mimics software released by Materialise:
`http://www.materialise.com/mimics`

from patient to patient. These all make the screws very difficult to be inserted accurately.

In common practice, surgeons must examine the 2D CT sectional images of a patient carefully before an operation to determine the screw insertion positions in three dimensions. Although specialists, such as radiologists, are trained to interpret 3D information from a series of 2D CT or MRI images, surgeons who perform operations often find it difficult to relate cross-sectional slices to actual 3D organs. [2] suggests a computerized system to provide intra-operative visual assistance. With this system, surgeons can manipulate CT images by rotating the slides to align them with the axis of a vertebra. They then plan the orientation and the length of the screw to be used. It has been shown that the failure rate of screw insertion without computer assistance varies between 28.1% and 39.9% [4].

Recent research shows the importance of Bone Mineral Densities (BMD) as a decisive element for screw insertion surgeries [3]. Apart from anatomic landmarks and nerve positions, the success of the surgery depends highly on whether the screw is inserted to an appropriate region whose densities are sufficient to "secure" the screw. A misplaced screw to regions of low BMD will lead to early loosening and breakdown of the structure, as well as other neurological complications [4]. With knowledge on how the BMD are distributed over the bone volume, screws are inserted more safely and accurately. This entails the capability to visualize clearly the 3D BMD distribution in a bone. Currently available commercial medical software provides certain degree of 3D visualization by displaying a 3D iso-surface model extracted from 2D medical images. One limitation of the iso-surface visualization is that it can only show surface details but not the internal structures of an object.

A joint research project has been carried out in cooperation with the Department of the Orthopaedic Surgery at The University of Hong Kong. The primary goal is to build a customized visualization system, VISBONE, for screw insertion planning using BMD information. The system incorporates advanced graphics techniques and its main functions shall include:

- supporting 3D interactive bone display,
- displaying volumetric BMD distribution,
- providing pre-surgical decision-making assistance, e.g. feasible path indications,
- simulating the screw insertion operation, and
- offering tools for post-surgical evaluation.

A prototyping system has been built at the current stage. Various pilot investigations have been carried out during the course of research studies. We shall discuss in this thesis the design and implementation of the experimental system, as well as some solutions we proposed in applying computer visualization techniques to pedicle screw insertion planning.

The remainder of this thesis is organized as follows. In Chapter 2 we shall give a brief overview of the background information on the project itself, particularly the origin of medical visualization. The state-of-the-art graphics techniques on volume rendering will also be mentioned. The general features supported by VISBONE will be described in details one by one in Chapter 3. We shall then discuss in Chapter 4 some specific functions or tools that are primarily designed for our application, namely, "Surgical Planning for Pedicle Screw Insertion". Finally, the thesis will be concluded in Chapter 5 where some possible future developments of the project are discussed.

Chapter 2

Background

2.1 From Surfaces to Volumes

If you search the words "3D models" from the web using any search engine, you may be overwhelmed by hundreds of search results referencing Internet sites that provide graphical models describing 3D objects. One of these examples is the famous Stanford bunny.[1] These models are made up of vertices, edges and facets. Strictly speaking, they are just 2D surface description of the topology or outlook of 3D objects. If you cut these graphical models into halves, you will find that the 2D surface is just like a crust enclosing an empty interior.

All graphics accelerator specialized in graphical display can handle the mentioned geometric primitives, i.e. vertices, lines and polygons, efficiently. Nowadays, people are talking about graphics accelerators that can process and render more than 30 million triangles in a second. Throughout the years, manufacturers have kept on improving their graphical hardware to meet the market's expectations. At the same time, many geometric and graphics algorithms have been developed for fast manipulation of surfaces. These all form the basis for surface-based rendering. This conventional mode of rendering is quite successful. Despite the fast polygon processing requirements for on-line games and real-time walkthroughs, surface-based rendering has made these applications becoming possible even on low-end personal computers.

Although surface-based rendering is well-developed and well-supported by hardware, it has its own limitations. Looking at some surface-based rendered images and you may find its disappointing performance in dealing with natural phenomena like cloud, smoke and fire. These are often solved by having photo-textured polygons instead. The lack of intrinsic 3D description for these natural phenomena is because of their inherent characteristics in nature. They cannot be simply described by the surface-based geometric primitives. They occupy volume in space which is not accountable by surface-based rendering. Besides natural phenomena, many applications may need to deal with volumetric data sets in

[1] The Stanford bunny is publicly available at
`http://www.cc.gatech.edu/projects/large_models/bunny.html`

the sense that they contain information for each location in space. Seismic, atmospheric, and medical data sets are some of the examples. Consider a data set describing the temperature variations of the atmosphere above Hong Kong. It contains the temperature values sampled at different longitudes, latitudes and heights above the sea level. This organization makes the data set almost impossible to be visualized using surface-based rendering. To get around this difficulty, one must impose some surface-based primitives to the data sets. [5] suggested an efficient algorithm in defining surfaces passing through data points of the same scalar value, in this example, a particular temperature. The surface so formed is called an iso-surface and can then be rendered efficiently.

Another attempt made by researchers for visualizing volumetric data sets is to seek directions totally different from the surface-based approach. Geometric primitives like points, lines, and polygons are abandoned but voxels are used instead. Voxels become the basic elements in the new approach called volume-based rendering.

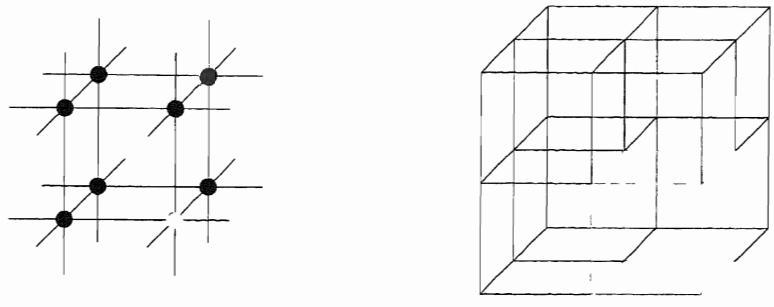
2.2 Volume Visualization

Volume visualization is the techniques of rendering and manipulating volumetric data sets. Among the various applications of volume visualization, medical visualization is one of the major category because of the vast amount of volumetric data sets coming from medical imaging. Before going to the details of medical imaging and medical visualization, let us first take a look at the basics for volume rendering.

2.2.1 Voxels

The basic element of a volume is a voxel. A voxel is analogous to a pixel in a 2D image. While each pixel is addressed by an (x, y) couple indicating the coordinates of two orthogonal axes, a voxel is addressed by an (x, y, z) tuple indicating the coordinates of three orthogonal axes. A voxel with address (i, j, k) is located at the i^{th} row, j^{th} column of the k^{th} slice. There are two different interpretations of a voxel. One saying is that a voxel is a grid point in a 3D lattice without occupying any space (Figure 2.1(a)). Another is to consider a voxel as a space-occupying rectangular block (Figure 2.1(b)). Many of these blocks are packed tightly to form a volume. In fact, these two different settings can be thought of as one, because in the second case, the centre of a rectangular block can be taken as a "grid point" and can then work similarly as the first.

Each voxel in a volume stores either a scalar value, e.g. temperature, or a vector value like wind direction. A volume of scalar values is called a *scalar field* while that of vector values is called a *vector field*. In this thesis, the word "volume" is used interchangeably with "scalar field" since the volumetric data sets we are focusing on throughout our studies encodes scalar values, e.g. bone densities, only.



(a) (b)

Figure 2.1: Voxel Representation. (a) As grid point; (b) As rectangular block.

A 3D scalar volume containing all the voxels can be considered as embedding a function $f : \mathbb{Z}^3 \rightarrow \mathbb{R}$. If the range of f is 0 and 1 only, or the voxels of a volume stores only 2 discrete values, the volume is called a binary volume. Otherwise, it is called a gray-scaled volume because of the different voxel values that each voxel may be associated with.

2.2.2 Rendering

Having defined the above volume settings, the question is how to display a volumetric data set. Volume rendering is the display of a 2D projected image of a 3D volume from a given viewpoint. In the seminal paper on volume rendering [6], Levoy derived the well-known compositing functions which accumulate the color and opacity of each voxel to form a 2D image. To determine the intensity of a pixel on the image plane, a ray is cast from the viewpoint through the pixel into the volume. Sampled points are taken along the ray whose intensities and opacities are accumulated to obtain the final intensity of the pixel.

Back-to-front Composition

Suppose there are n sampled points $x_0, x_1, \ldots, x_{n-1}$ along a ray. Let the color and the opacity for the i^{th} sampled point be $C(x_i)$ and $\alpha(x_i)$ respectively. A *back-to-front composition* treats the ray as one that comes from the background, passing through the data volume, and then reaches the viewpoint. Consider when the ray passes through x_i (Figure 2.2). The color or intensity of the ray $C_{in}(x_i)$, after passing through $x_0, x_1, \ldots, x_{i-1}$, is attenuated by the opacity of x_i. The resulting composition $C_{out}(x_i)$ is the accumulation of the attenuated intensity and the color of x_i moderated by its own opacity. The compositing function is thus as follows:

$$C_{out}(x_i) = C(x_i)\alpha(x_i) + C_{in}(x_i)(1 - \alpha(x_i)) \tag{2.1}$$

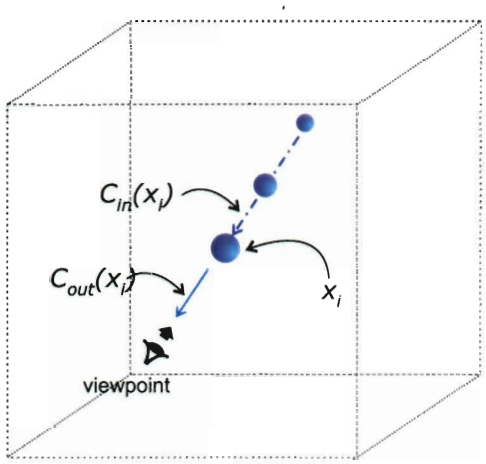


Figure 2.2: Compositing the i^{th} sampled points along a ray.

With iterations on $i = 0, \ldots, n-1$, $C_{out}(x_{n-1})$ is the final composition of the pixel where the ray passes through the image plane. Notice that $C_{in}(x_0)$ can be set to any value representing the background color.

Equation 2.1 is known as the back-to-front compositing formula suggested by Levoy to approximate the light attenuation phenomena of viewing a volumetric data set. The formula can be modified such that a totally different rendering is obtained. For instance, instead of accumulating the colors and opacities of all sampled points along a ray, the maximum (or the minimum) intensity among the sampled points along a ray can be taken as the output intensity. This is called *Maximum* (or *Minimum*) *Intensity Projection* which is a common technique for visualizing blood vessels (Figure 2.3).

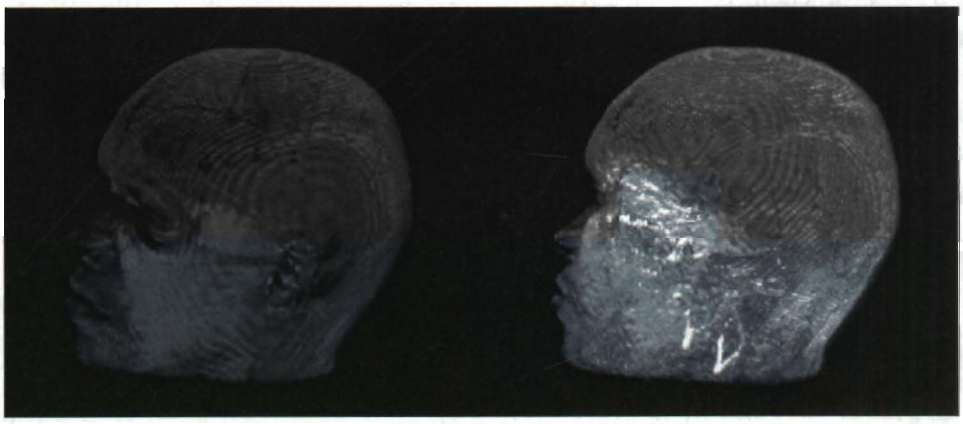


Figure 2.3: Different compositing functions. Left: Levoy's accumulation. Right: Maximum intensity projection. Note the blood vessels near the bottom of the image on the right.

Tri-linear Interpolation

During the composition, if a sampled point does not coincide with a voxel, interpolation technique, such as the tri-linear interpolation, is applied to the voxels surrounding the sampled point to obtain an estimation of the intensity and opacity of the sampled point.

Tri-linear interpolation is an extension of the simple linear interpolation to the three dimensions. The value v_x at position $x \in [p_0, p_1]$ is obtained by a linear interpolation of values v_0 and v_1 at locations p_0 and p_1 respectively and is given by:

$$v_x = \frac{x - p_0}{p_1 - p_0}(v_1 - v_0) + v_0$$

We denote this linear interpolation by $v_{01} = \Lambda_{0,1}(x)$. A superscript x, y, or z is used to represent an interpolation along a particular axis direction when 3D point locations are involved.

To obtain an interpolated voxel values for a sampled point (x_0, y_0, z_0), we need to consider its surrounding 8 voxels that form a cube (Figure 2.4(a)). Three passes of linear interpolations are performed, one for each of the three volume axes, x, y, and z. Four linear interpolations along the x direction are first carried out to produce intermediate voxel values: $v_{01} = \Lambda^x_{0,1}(x_0)$, $v_{23} = \Lambda^x_{2,3}(x_0)$, $v_{45} = \Lambda^x_{4,5}(x_0)$, and $v_{67} = \Lambda^x_{6,7}(x_0)$ (Figure 2.4(b)). The second pass consists of the two linear interpolations along the y direction: $v_{0123} = \Lambda^y_{01,23}(y_0)$ and $v_{4567} = \Lambda^y_{45,67}(y_0)$ (Figure 2.4(c)). Then, the value (x_0, y_0, z_0) is given by one final interpolation along the z direction: $v_{01234567} = \Lambda^z_{0123,4567}(z_0)$ (Figure 2.4(d)).

Speed-ups

The ray-casting approach requires intensive computation for rendering a volume. Its view dependent nature implies that each time the viewpoint is changed, the computation will have to be carried out again. Interactive rendering is therefore difficult to achieve. Various methods have been proposed to reduce the rendering time. In [7], Laur and Hanrahan describe a progressive refinement algorithm which is based on hierarchical splatting. Gouraud-shaded polygons are used to approximate the splats. Since graphics workstations are optimized to draw polygons, the rendering time is improved.

Rendering a subvolume instead of the entire volume also speeds up the frame rate. In [8], Shin et. al. suggest that only a thin slab within the volume is drawn. In this way, not only the rendering is made faster, occluding organs and structures are also removed.

Apart from algorithmic solutions, the problems of slow rendering and huge data sets can also be overcome by the availability of increasing memory capacity and the graphics chipsets [9] which are designed specifically for volume rendering. These together have made possible the migration of the computation-intensive components in the volume rendering pipeline down to the hardware level.

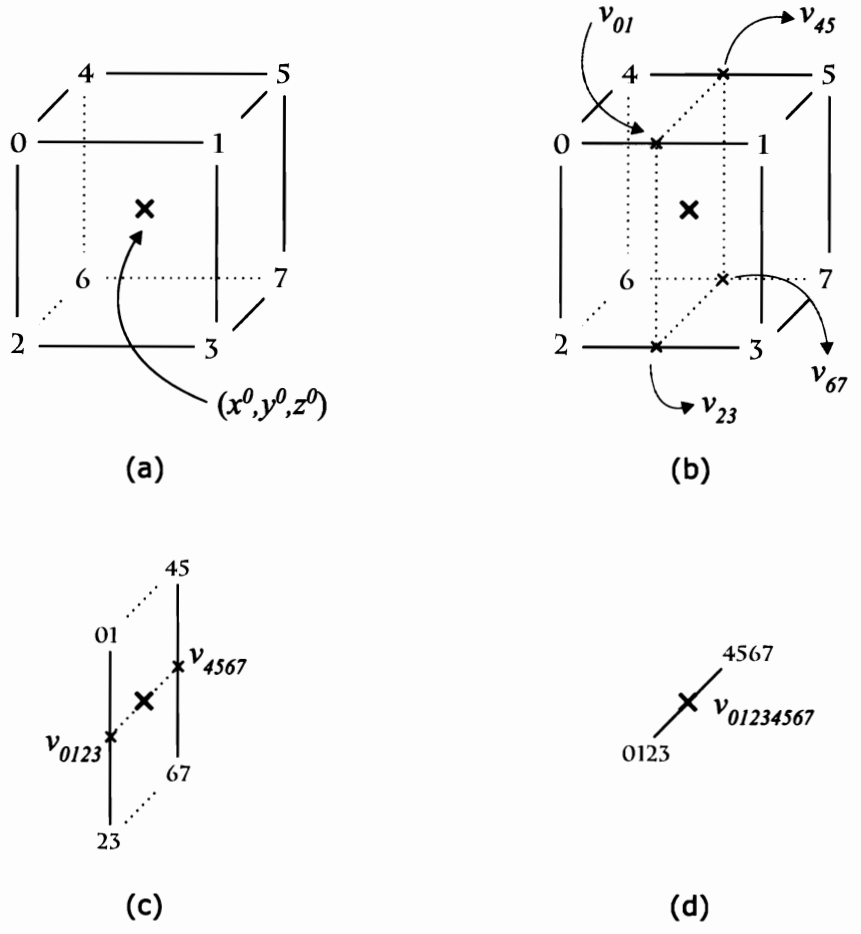


Figure 2.4: Tri-linear interpolation. (a) 8 neighbouring voxels of a sampled point; (b), (c), and (d) Linear interpolations along x, y and z respectively.

Hardware accelerated rendering

Accelerated volume rendering algorithms using 3D texture mapping hardware are proposed in [10, 11]. A volume is treated as a 3D texture image which is loaded into the texture memory. Parallel planes passing through the texture space are then drawn. The 3D texture is mapped to the planes by supplying appropriate texture coordinates for the vertices of the polygons. Voxel values are mapped to different opacities and colors using a Look Up Table (LUT). Direct volume rendering is done by blending the planes back-to-front into the frame buffer using a specifically chosen alpha blending function.

The parallel planes are drawn either aligning to the image plane or to the volume axis. In both cases, opacities and colors of the regions in-between the texture voxels are found by tri-linear interpolation of adjacent voxels. Interpolations are done by the graphics hardware which is capable of performing the

tedious calculations in real-time. This hardware feature has made interactive volume rendering becoming possible.

2.2.3 Advanced Volume Manipulations

Besides hardware enhancement, graphics library may also provide effective operations for volume data. Recently, Westermann and Ertl [12] propose several efficient techniques to edit volume data with OpenGL and its extensions. One of the techniques is the use of OpenGL stencil buffer tests to perform clipping by arbitrary geometries (OpenGL only supports clipping by means of planes). Any close surface can be a clipping geometry. To enable clipping, we only need to determine the area of intersection between the clipping geometry and each of the textured polygons used for rendering the data volume. The stencil buffer is used to lock this area so that drawing into the frame buffer is disabled within this area. We do this for each of the polygonal slices, and a hollow clipped volume is formed.

To determine the area of intersection between the clipping geometry and a textured polygon, the authors suggest that a clipping plane be placed at the same position and orientation as the textured polygon. Only back-facing fragments of the clipping geometry that are behind the clipping plane with respect to the viewpoint are rendered (with the frame buffer being disabled) and the stencil buffer be set in these regions. However, there may exist back-facing fragments that are not within the intersection area but still are rendered (Figure 2.5(b)).

In fact, these back-facing fragments falling out of the intersection area must have some front-facing fragments covering them behind the clipping plane. By drawing the front-facing fragments behind the clipping plane and clearing the stencil buffer in these regions, the stencil buffer is left with regions that have been set with a special value which signify the intersection of the clipping geometry and the textured plane (Figure 2.5(c)).

This method has been adapted in our system with slight modifications in the process of determining pedicle screw insertion feasibility and will be discussed in details in section 4.3.1.

2.3 Medical Imaging

People realized long time ago the importance of acquisiting the internal structure of the human body for medical studies. The study of cadaver anatomy was followed by the making of anatomical atlas of human bodies. These illustrations formed the basic human body navigation maps for all surgical operations at that time. As the famous anatomic paintings by Leonardo da Vinci have shown (Figure 2.6), there was a need for sophisticated and clearly presented fine details rather than primitive hand-sketching. Moreover, it would be impossible to have illustrations for living men. This demand indeed pushed the evolution of medical imaging, starting from X-ray discovered in the mid-19^{th} century. Since then,

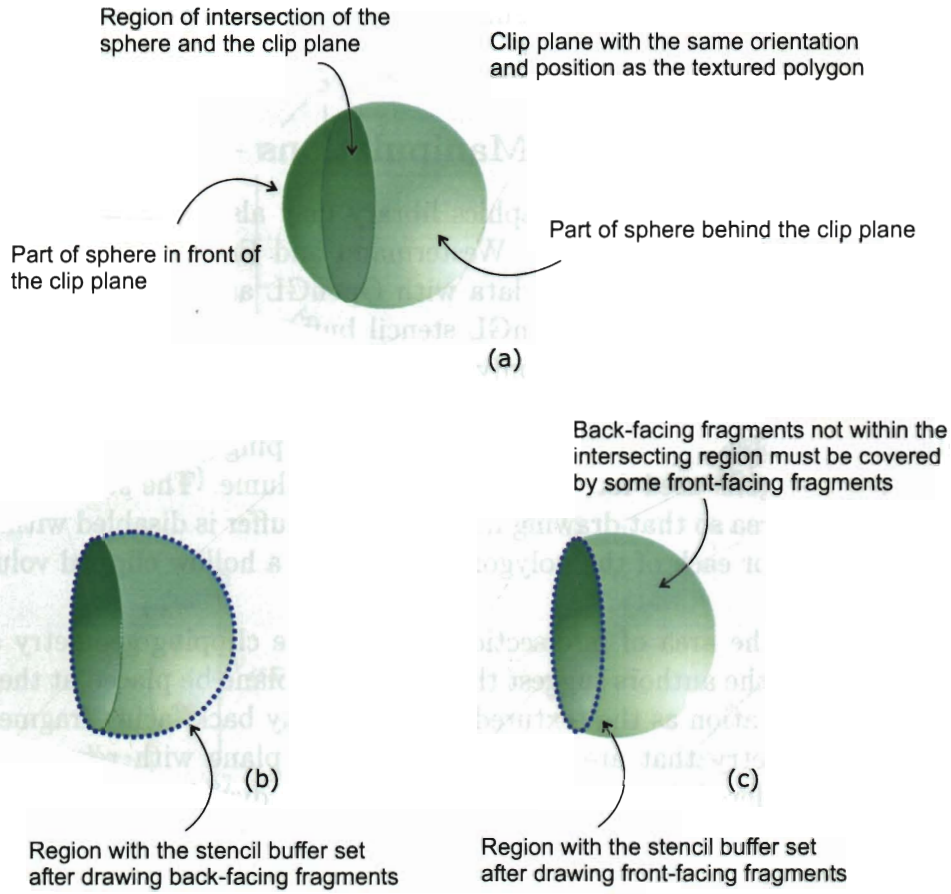


Figure 2.5: Arbitrary clipping geometry. A sphere is used as the clipping geometry for illustration. (a) A clipping plane is set up with the same orientation and position as the textured polygon intersecting the sphere. (b) After drawing back-facing fragments, the region with the stencil buffer set contains fragments not falling within the intersection. (c) Drawing front-facing fragments then will correct the region marked by the stencil buffer.

different medical imaging modalities had been developed. They are distinguished by their underlying image acquisition principles and properties. Each of them has its strengths and weaknesses against different tissues or diseases. Among these, Computed Tomography (CT) is the most commonly used imaging modality for producing bone-related medical images. To understand how it works, we first take a look at its origin, X-ray.

2.3.1 X-ray

X-ray was discovered by Wilhelm Konrad Röntgen in 1895. It was so named because Röntgen at that time considered it as a mysterious substance. X-ray has

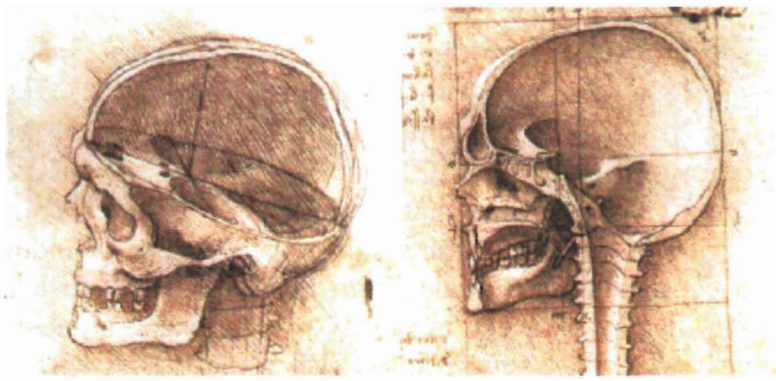


Figure 2.6: Famous paintings by Leonardo da Vinci of an anatomy of the skull. (From *Drawings of Leonardo da Vinci*, `http://banzai.msi.umn.edu/~reudi/leonardo.html`)

a very special penetrative nature. Its penetration power varies among different tissues. For example, bones absorb a greater extent of X-ray relative to muscles. When projecting a beam of X-ray through the human body to an X-ray sensitive photographic plate, shadows of different degrees are formed. This provided the first mechanism for taking photos of the internal structures of living men. In fact, Röntgen was awarded the first Nobel Prize in Physics in 1901 because of this mysterious X-ray he discovered.

X-ray is nowadays commonly used in various kinds of medical diagnosis. However, it is seldom used as data source for computerized medical visualization because of its lack of digital representation.

2.3.2 Computed Tomography

The first CT scanner was introduced by Sir Godfrey Hounsfield in 1972. CT is also termed "multi-pass X-ray" as compared with ordinary X-ray. To produce a single CT slide, an X-ray tube rotates around the scanned body on the plane to be imaged. At the opposite side of the body, a set of X-ray sensitive receivers connected to a computer records the X-ray intensities penetrating through the body. Measurements are taken as the tube rotates about 360° in one scan. After that, the computer performs a series of mathematical calculations called "backward reconstruction". The reconstruction then produces an array of values where each value represents the density at a particular location or pixel of the scanned plane. These values are called the CT numbers or the Hounsfield numbers in memorial of the inventor of this imaging technique. Hence the CT numbers are related to the densities of the scanned tissues. It is in particular useful for the study of the bone tissues since the Bone Mineral Densities (BMD) can be derived from CT numbers. As mentioned in Chapter 1, BMD is an essential element for making screw insertion decisions in bone surgeries.

Since these CT numbers are the result of some mathematical calculations performed by computers, they can be stored in digital forms. A single scan on a plane gives an array of two dimensional pixels describing tissue densities. Successive scans give a volume of density values in units of voxels. Hence CT data is very suitable, and in fact a rich source, for volumetric medical visualization.

One of the advantages of using CT scans is the application of a technique called "windowing". This means that we can control the contrast and hence highlight the region of interest in the resulting CT images. Windowing is done by assigning a continuous gray scale, from black to white, to a range of CT numbers. When displaying a CT image, tissues with CT numbers within the range will appear with the assigned gray level while tissues with CT numbers below and above the range will appear as black and white respectively. By using different window settings, different images highlighting different tissues are made from the same scan. Windowing and its application in visualizing CT data will be discussed again in details later in section 3.2.2.

Although we have mentioned several favourable aspects for CT imaging, there are yet downsides of it which make people hesitate in taking CT scans. As mentioned previously, CT scans are actually multi-pass X-ray imaging. Since X-ray is penetrative and hence radioactive in nature, excessive exposure to X-ray is harmful to human body. Scanning a single CT slide takes about 10 to 15 seconds. Having a vertebrae scanned needs 20-30 slides at 3mm separation, meaning that a patient is subjected to X-ray exposure for no less than 3 minutes. If high quality imaging is needed, the scanning time is even more. For patients undertaking surgeries, the number of CT slides required is minimized so that they are just enough for the surgeons' interests., e.g. an vertebrae. However, in most of the time we would like to study the entire structure, e.g. the spine, for research purposes. Hence it is very difficult to obtain CT data sets for such occassions. Nevertheless, we shall use CT data here for the entire study of 3D bone visualization.

2.4 Medical Visualization and Computer-Assisted Pedicle Screw Insertion

Early applications involving visualizing medical data were mostly surface-based. In [13, 14, 15], contour lines are first extracted on each 2D image representing the outline of an organ or a structure, e.g. face. This is a very common approach in orthopaedic surgery since the bone surface can easily be distinguished from the relatively soft muscle tissues. After extracting the contours, triangular patches are formed between contours from successive images (Figure 2.7). A surface bone model is then formed and displayed in real-time.

[2] was among the first to introduce computer assistance into pedicle screw insertion surgery. This work is focused on intra-operative guidance. The system is 2D CT image-based. It provides screw insertion guidance by complicated

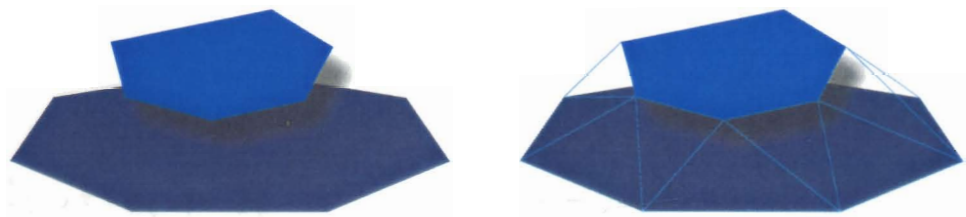


Figure 2.7: Surface reconstruction from contours. (a) Two polygons representing contours extracted from two adjacent images; (b) Surface patches connecting the two contours.

matching of anatomic landmarks in CT images and images taken during the operation. This process is called registration. A space pointer can then be moved around and its corresponding location in the CT image is shown. Clinical studies claimed that the failure rate of pedicle screw insertion surgeries with the assistance of such systems is reduced significantly from 39.5% down to 2.5%.

Based on these clinical studies, commercial products were developed with cameras and electronic devices attached to the operation bed for the registration process. One of the drawbacks is that these devices may hinder the actions of the surgeons. Moreover, the success of the system highly depends on the accuracy of the registration. If the positions of the cameras or detecting devices are altered accidentally, the registration process has to be restarted all over again. Or to the worst, if the changes are not discovered in time, the surgeons will be falsely informed and the patients will be subjected to higher risks.

Up till now, there are no visualization tools addressing the importance of Bone Mineral Densities (BMD) in pedicle screw insertion surgeries. Although it would be safe to insert a screw by totally embedding it into the pedicle, the BMD distribution of the pedicle plays an important role for the sustenance of the implanted structure. If a screw is inserted into a low density region, it will probably experience early loosening and the structure will collapse shortly after the surgery. When this happens, the patient will have to undergo the surgery and suffer from risks and pain once again.

Although the screw must be inserted into a confined region around the pedicle, there are still flexibilities in choosing the orientation of a screw within the pedicle so that it is secured into regions with higher Bone Mineral Densities. If the surgeons are provided with the BMD information, they can choose the most appropriate direction for inserting the screw to ensure that the resulting implant is the most long-lasting one. In view of this, the major aim of this research is to design a 3D visualization system which provides BMD information for pedicle screw insertion surgeries. The system will incorporate advanced graphics visualization techniques to meet the specific requirements posed by these kinds of

surgical operations.

Chapter 3

VISBONE

A joint project is conducted between the Department of Orthopaedic Surgery and the Department of Computer Science and Information Systems at HKU. The aim of the project is to develop a customized computer visualization system, called VISBONE, to enhance surgical operation planning for pedicle screw insertion.

By combining successive CT slices, a 3D bone volume is formed. This volume encodes the same piece of information as the original 2D slices, e.g. BMD values. Using direct volume rendering accelerated by 3D hardware texture mapping to achieve interactive frame rates, a 3D virtual bone can then be manipulated interactively. This direct 3D volume visualization, together with other computer graphics volume visualization techniques, can assist surgeons in making the decisions for screw insertions, in addition to the 2D CT images examination. A prototyping system has been built which integrated several special features for the aimed tasks. These include cutting through the bone at any arbitrary orientation, distance and slab thickness measurements, BMD thresholding and partitioning. The current system also allows a simulation of screw insertion at any insertion point. Once the insertion location is fixed, the system provides the BMD information along all possible screw orientations. Surgeons can then decide which is the best direction for inserting the implants.

In this section, we shall describe in details the special requirements and the system implementation of the general features for VISBONE.

3.1 System Requirements

As VISBONE is a customized system designed specifically for the planning of pedicle screw insertion surgeries, there are some special requirements uniquely spelled out by the application. These are also the general expectations of surgeons who would like to make use of VISBONE to assist in their screw insertion decision-making processes. The requirements are as follows:

- SUPPORT INTERACTIVE DISPLAY. Surgeons do not have the time to wait long before they can obtain the visualization results after issuing any command.

- USER-FRIENDLY INTERFACE. Tools for manipulating the virtual bone are kept easy to understand and use.
- SHOW OPERATION-RELATED INFORMATION. Information like slab thickness or point-to-point distance is important to surgeons for selecting a screw with appropriate length.
- SHOW BMD DISTRIBUTIONS. Bone Mineral Densities distributions are the major concern for the surgeons to ensure that the inserted region can securely hold the screw.
- INDICATE SCREW INSERTION PATH FEASIBILITY. The system is expected to be able to show which orientations at a particular bone surface point are feasible for screw insertion.

VISBONE runs on an SGI Onyx2 workstation equipped with an Infinity Reality2 graphics board, MIPS R10000 CPU, 512 Mbytes of main memory and 16Mbytes of hardware texture memory. The workstation is able to perform hardware texture tri-linear interpolations. Hence it is chosen as the underlying platform so that VISBONE can perform interactive volume rendering using accelerated hardware texture mapping.

3.2 Data Set Preprocessing

3.2.1 Image Resampling

Each CT slice is a 2D image of resolution 512×512 pixels. Since many of these slices are fed into the texture memory for rendering, measures have been taken to reduce texture memory consumptions. We found that we can reduce the resolution of these images to 128×128 pixels without significant loss of visual details. By doing this, the texture size is reduced to $\frac{1}{16}$ of the original size.

To reduce the resolution of the CT images, the most straight-forward way is to throw away every 15 out of 16 pixels. Hence, only 1 pixel out of a block of 4×4 pixels is left in the low-resolution image (Figure 3.1).

Although this method is easy to implement, it misses out some important details of the original image. Consider if there is a fine line featuring an important structure in the original image, it is highly possible that the line is dropped out because it does not coincide with the lucky pixel that we keep for each block (Figure 3.1(b)). If this happens, the fine line will totally disappear in the resulting image. Hence we have to be careful to take into account these situations for image resampling.

Here, we borrow the idea of neighbourhood averaging from image processing techniques [16]. The idea is commonly used in smoothing out noise in images. In the neighbourhood averaging method, each pixel is replaced by the average of itself and the $n^2 - 1$ neighbours in a square block of size $n \times n$. Our approach is to combine the averaging and the drop-out processes. Each block of 4×4 pixels

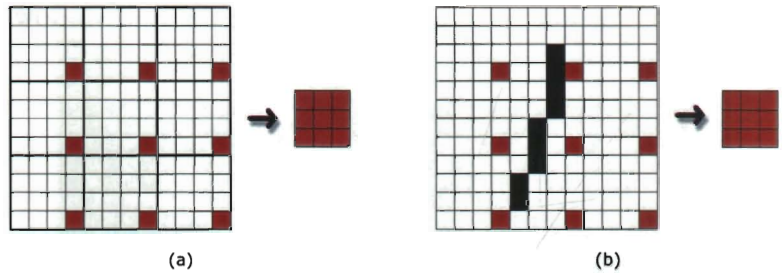


Figure 3.1: Down-sampling an image (from 12 × 12 to 3 × 3 by the drop-out method. (a) only 1 pixel (in brown) out of a 4 × 4 block is left. (b) the reduced image omits a fine line (in black).

is replaced by one single pixel representing the average value of the 16 pixels within the block. This reduces the image resolution to an acceptable level while retaining special image features that might otherwise be omitted in the purely drop-out process. Figure 3.2 shows the image quality of reduced resolution using the two methods.

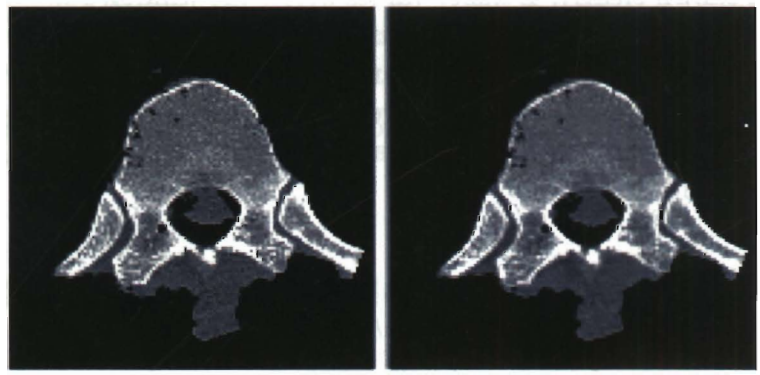


Figure 3.2: Comparison of image quality for down-sampled images. Left: By the drop-out method; Right: By neighbourhood averaging. The two images are the 128 × 128 versions of a 512 × 512 original.

3.2.2 Windowing

After the CT scanning process, a CT image is produced which contains the array of CT numbers resulting from the backward reconstruction calculations (section 2.3.2). Different intensity levels are assigned to the pixels according to the CT numbers. A CT image is thus formed with varying intensities that are directly related to the densities of the imaged tissues. To enhance the visualization of the CT images, radiologists use a *window* to define a range of CT values (or density

values since they are directly related to each other), to highlight the tissues of interest contained in CT images (Figure 3.3). Within the window range, den-

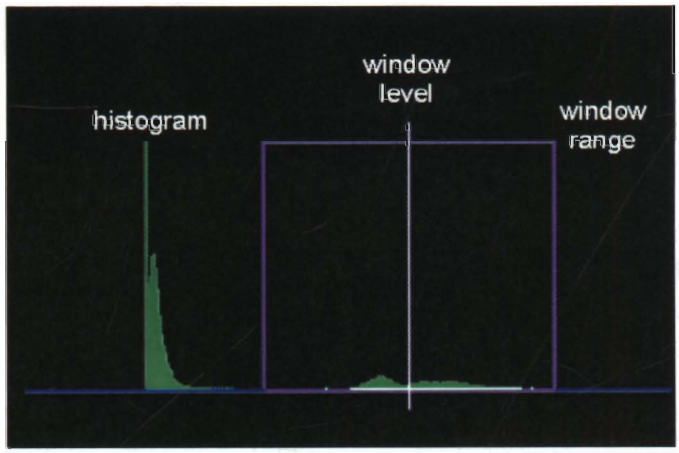


Figure 3.3: Windowing: intensity levels will only be assigned to density range within the window range centered at the window level.

sity values are assigned more intensity levels so that minute density variations are magnified and shown clearly. Density values below the window range are assigned the minimum or zero intensity while density values above the window range are assigned the maximum intensity. A linear function is used to map density values within the window range to intensity levels. If intensity levels range from 0 and 1, the following function can be used to assign a linear intensity levels according to the density values:

$$g(x) = \begin{cases} 0, & \text{if } x < wl - wr; \\ \frac{x - wl + wr}{2wr}, & \text{if } wl - wr \leq x \leq wl + wr; \\ 1, & \text{otherwise.} \end{cases}$$

where wl and wr are the window level and window range respectively.

To highlight bone tissue but suppressing soft tissue details, a bone window is defined which encloses the densities representing bone regions as shown in Figure 3.4(top). A different window setting is used to magnify the contrast for visualizing soft tissues (Figure 3.4(bottom)).

We have described how to apply window setting to 2D CT images during the data preprocessing stage. This step can in fact be performed at a later stage, where the window setting is applied directly to the 3D volume data. We choose not to do so because of the specific purpose of this application, namely, the visualization of bone structures. In comparison with other general visualization tools, a static window setting capturing the possible range of bone densities is sufficient for our needs. Dynamic window settings is only desirable when an application wants to highlight structures of different density ranges.

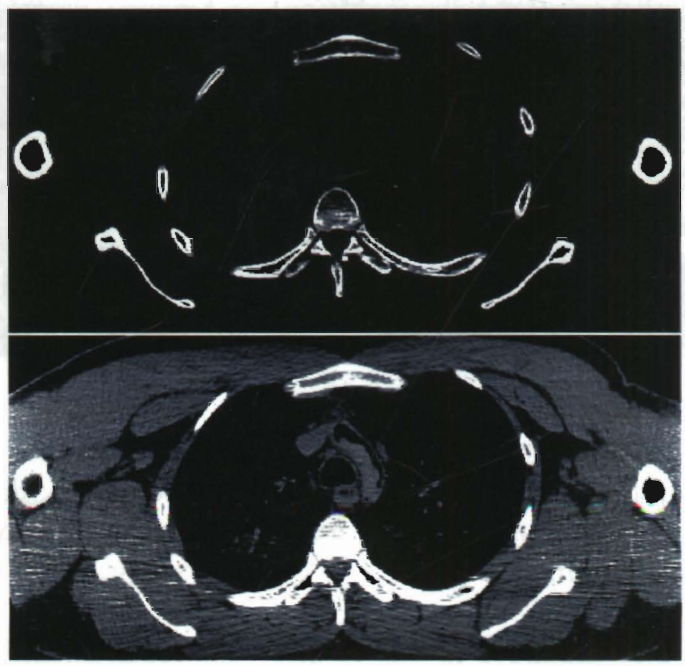


Figure 3.4: Effect of different window settings. Top: Bone window; Bottom: Soft tissues window. The slice shown is taken from the Visible Human CT data set.

The series of CT slices, after the process of resampling and windowing, are then piled together to form a volume which is loaded into the system as a 3D texture.

3.2.3 Histogram Equalization

Image enhancement is necessary in many applications where image quality is the primary goal. We tried to apply histogram equalization to the window defined on the 2D CT slices. Although the quality of the final rendering has been enhanced (Figure 3.5), it is then relatively hard to tell the differences in the densities based on color separation shown on the color scale. This is because histogram equalization does not result in a linear mapping from density values to voxel values, which are then used to determine the voxel intensities. Hence, we suggest that it is not appropriate to use histogram equalization since the resulting image does not truly reveal the density distribution.

3.3 Rendering Settings

The data volume, after the preprocessing stage, is treated as a 3D volumetric texture and loaded into the texture memory. Due to hardware limitation, the size of each dimension of the texture volume must be a power of two. Hence, the

Figure 3.5: Image qualities of windowing, Left: with; Right: without, histogram equalization. Although the image is enhanced by histogram equalization, the image does not truly reveal the density distribution because of the non-linear mapping from density values to voxel values.

data volume is first expanded in size to meet this requirement. Regarding the viewport or window size, we found that the larger the window size, the slower the frame rate. This is because for large window size and hence a greater number of pixels within the window, the amount of tri-linear interpolations that has to be carried out will increase. As a result, the window size is fixed at 500×500 pixels which offers sufficient visual details while maintaining reasonable interactivity.

3.3.1 Drawing polygons

To render the data volume, parallel planar polygons passing through the texture space are drawn. Textured images are mapped to the polygons by assigning appropriate texture coordinates to their vertices as if the polygons are cutting through the data volume. The textured polygons are then rendered in a back-to-front manner. Each polygon with different opacities and colors at different points on the plane is blended into the frame buffer. This is to simulate the direct ray-casting approach for compositing the final volume visualization.

Image plane aligned vs. texture axis aligned polygons

There are two ways of drawing the parallel planes. The first one is to draw planes parallel to the image plane, i.e. *image plane aligned*. When the data volume rotates, the planes may then cut the texture volume at any arbitrary angle (Figure 3.6). Opacity and color of the empty space in-between texture voxels is filled in by the graphics hardware using tri-linear interpolations (section 2.2.2).

The second method is to draw parallel planes along the direction of any of the three texture axes, i.e. the planes are always parallel to either the xy-, yz- or zx-plane of the texture volume. We called this *texture axis aligned*. Depending on the current orientation of the texture volume, the axis plane with the largest

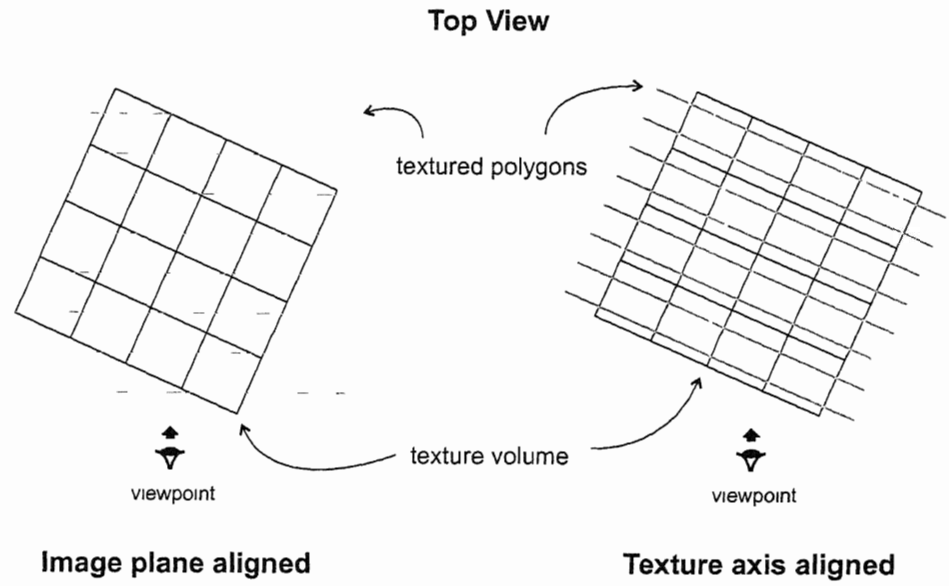


Figure 3.6: Image plane aligned vs. texture axis aligned textured polygons.

area of projection to the image plane is chosen. Texture mapped planes are then drawn parallel to this axis plane. The determination of the axis plane will be discussed later in this section.

The two methods have the same visualization output if the underlying graphics system supports 3D hardware texture mapping. Both methods have been tried on VISBONE. 10 data sets varying in size are used in the testing (Table 3.1). The frame rates of rendering the datasets in a window of size 500 × 500 pixels

Table 3.1: 10 data sets with different data sizes.

Data Set	Data Volume		Texture Volume	
	Dimension	Size (Bytes)	Dimension	Size (Bytes)
1	5 × 5 × 30	750	8 × 8 × 32	2048
2	19 × 19 × 20	7220	32 × 32 × 32	32768
3	19 × 19 × 40	14440	32 × 32 × 64	65536
4	39 × 39 × 80	121680	64 × 64 × 128	524288
5	128 × 128 × 64	1048576	128 × 128 × 64	1048576
6	128 × 128 × 128	2097152	128 × 128 × 128	2097152
7	128 × 128 × 157	2572288	128 × 128 × 256	4194304
8	256 × 256 × 55	3604480	256 × 256 × 64	4194304
9	512 × 512 × 12	3145728	512 × 512 × 16	4194304
10	256 × 256 × 128	8388608	256 × 256 × 128	8388608

using the two methods are shown in Figure 3.7.

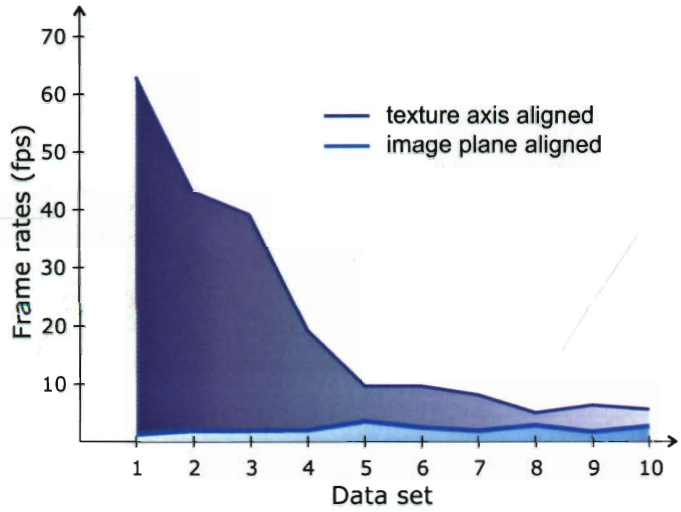


Figure 3.7: Frame rate comparison of the two different plane drawing approach: *image plane aligned* and *texture axis aligned.* Window size is 500 × 500 pixels.

From the experimental results, we found that the frame rate for rendering a volumetric data set using the image plane aligned polygon rendering method is on average about 2 frames per second. This can hardly constitute an interactive rendering. If we use the texture axis aligned method, an average frame rate of about 10 fps can be achieved. The reason for the discrepancies between the two methods is that for the image plane aligned approach, intensive calculations have to be done for the tri-linear interpolations. However, interpolation computations for the texture axis aligned approach is less complicated and the approach hence offers a higher frame rate. Since the two methods give identical visual results, we choose the texture axis aligned approach for VISBONE as the basis for rendering.

Maintaining a constant frame rate

One polygon is drawn for each voxel slices in the volume so that each voxel will contribute to the composition. For instance, when the polygons are parallel to the xy-plane, the number of polygons drawn is equal to the number of voxels in the z dimension. However, this poses a problem of variating frame rate. The number of polygons drawn may keep on changing when the view direction alters because the three volume dimensions may not be the same. Using data set 9 as an example, we can see from Figure 3.7 that it has a higher frame rate of 10.08 fps than the larger data set 10. The data set has a small z-dimension of 16. The overall average frame rate is out-balanced by the fast rendering in this view direction.

To correct for the frame rate inconsistency, we fix the number of polygons drawn to be a constant. The maximum volume dimension is taken as the number

of polygons to be drawn. For a particular viewing direction, some "dummy" polygons are drawn, outside of the data volume, if the number of polygons is greater than the volume dimension in that direction (Figure 3.8). This will

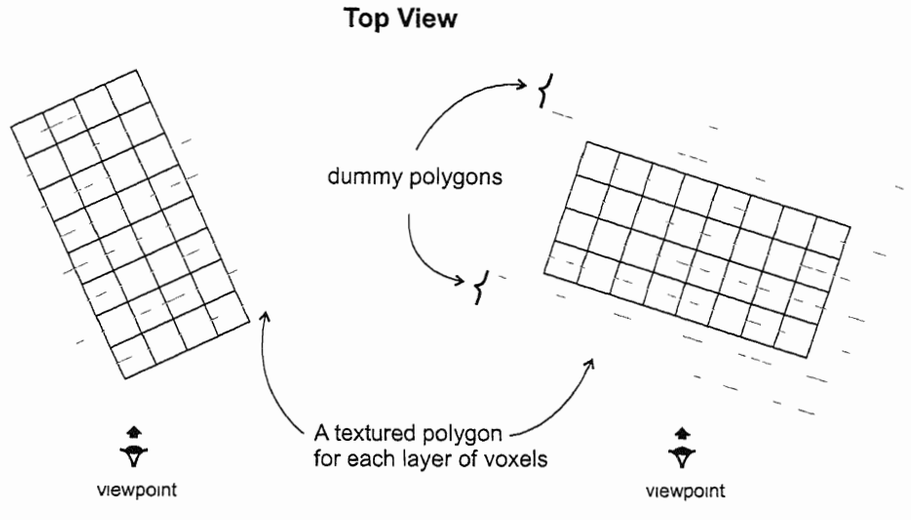


Figure 3.8: Dummy polygons added to maintain constant frame rate.

compensate for the drastic changes in frame rates because of the difference in the three volume dimensions.

Determination of Axis Plane

The orientation of the texture volume is given by a rotational matrix R, where

$$R = \begin{bmatrix} r_{00} & r_{01} & r_{02} \\ r_{10} & r_{11} & r_{12} \\ r_{20} & r_{21} & r_{22} \end{bmatrix}.$$

Assume we are looking into the scene along the negative $\vec{z}$ world direction (Figure 3.9). We first want to find out which direction $\vec{p}$ w.r.t. the texture volume is now pointing at the $-\vec{z}$ world direction. Hence, we have

$$R \cdot \vec{p} = -\vec{z},$$

or

$$\vec{p} = R^{-1} \cdot -\vec{z}.$$

Suppose that R^{-1} is given by

$$R^{-1} = \begin{bmatrix} \tilde{r}_{00} & \tilde{r}_{01} & \tilde{r}_{02} \\ \tilde{r}_{10} & \tilde{r}_{11} & \tilde{r}_{12} \\ \tilde{r}_{20} & \tilde{r}_{21} & \tilde{r}_{22} \end{bmatrix}.$$

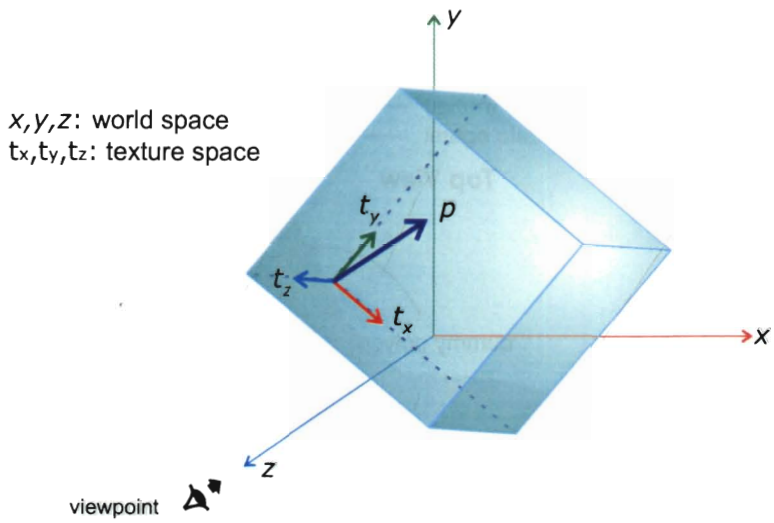


Figure 3.9: Axis plane determination. The vector $\vec{p}$ in texture space corresponds to $-\vec{z}$ in world space.

Since $-\vec{z} = (0, 0, -1)^T$, we have $\vec{p} = -(\tilde{r}_{02}, \tilde{r}_{12}, \tilde{r}_{22})^T$. We then proceed to decide onto which axis the vector $\vec{p}$ has the greatest projection by comparing the three components of the vector. The component (x, y or z) with the largest magnitude is chosen. The plane orthogonal to this axis is selected as the axis plane parallel to which the textured polygons are drawn. The sign of the selected component is also important for us to determine the order of drawing the polygons to ensure a back-to-front composition.

Although the mathematics presented here is rather straight forward, one may be aware of the heavy computation imposed by the need of computing R^{-1}. However, if we bare in mind that the rotational matrix R is indeed an orthogonal matrix, we shall immediately notice that the inverse of R is just the transpose of it, i.e. $R^{-1} = R^T$. With this remark, we can get rid of the matrix inverse computation that has to be carried out each time whenever the orientation of the texture volume is changed. Instead of having to solve for R^{-1}, we now have $\vec{p} = -(r_{20}, r_{21}, r_{22})$ with all three components being taken from R directly.

3.3.2 Setting up transfer functions

A bone volume contains voxel values that are directly related to the bone mineral density (BMD) of the bone tissues. To enable the visualization, transfer functions have to be set up that associate each voxel with opacity and color values according to its density. In short, a transfer function is one which maps the density values to other voxel attributes like opacity and color.

Transfer functions can be implemented using Look-Up Tables (LUT). LUT contain the opacity and color values for the possible range of voxel values. When-

ever a voxel is referenced, its opacity and color are retrieved immediately from the table by referring to the entry corresponding to the voxel's density value. OpenGL supports texture color mapping which transfers a texture value to RGBA components by referencing a look-up table. Although the look-up entries are specified in discrete density levels, values in-between adjacent entries are interpolated automatically.

A transfer function can be set up, for instance, to linearly interpolate RGBA values from $(0, 0, 0, 0)$ to $(1, 1, 1, 1)$ for the entire density range (Figure 3.10). Hence, voxels with the lowest density are assigned a black and totally transparent

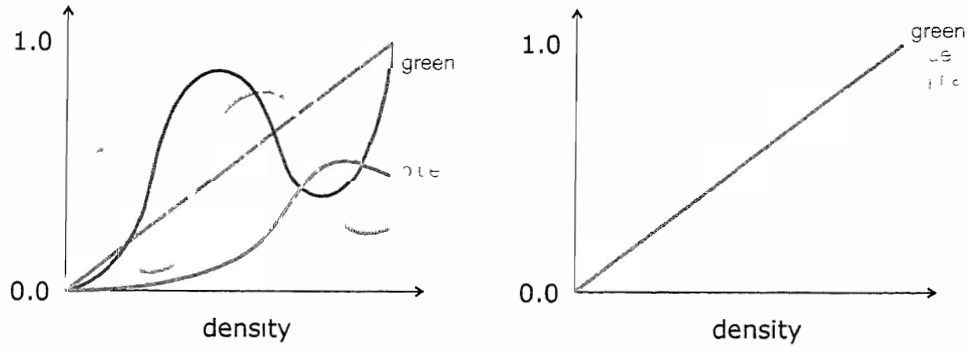


Figure 3.10: Transfer function. Left: Assigning (r, g, b, a) values to the density range; Right: A linear interpolation from $(0, 0, 0, 0)$ to $(1, 1, 1, 1)$.

appearance while voxels with the highest density values will appear as opaque white. Voxels with middle range densities will appear as semi-translucent grey. By using different independent functions on each of the red, green, blue and alpha components of the LUT, different special visual effects are achieved.

3.3.3 Composition

In volume visualization using hardware texture mapping, the process of accumulating and compositing opacities and colors along a ray in the ray-casting approach is simulated by opacity and color blending using the frame buffer.

Equation 2.1 in section 2.2.2 has set up the back-to-front composition for direct volume rendering. To relate the equation to the polygon rendering composition, consider the state when the i^{th} polygon is to be drawn. Drawing a polygon to the frame buffer is similar to an iteration of compositions for all rays. The $C_{in}(x_i)$ components of all rays are the pixel values currently existing in the frame buffer since they are the results of intermediate compositions of the previously drawn $i-1$ polygons. The $C(x_i)$ components are the pixel values that are blended to the frame buffer by drawing the i^{th} polygon. The resulting composition is controlled by setting up an appropriate blending function. The OpenGL library call `glBlendFunc()` when used in the following way:

```
glBlendEquationEXT(GL_FUNC_ADD_EXT);
glBlendFunc(GL_SRC_ALPHA, GL_ONE_MINUS_SRC_ALPHA);
```

gives the following composition:

$$C_d = C_s * A_s + C_d * (1 - A_s)$$

where C_d is the frame buffer pixel value, C_s and A_s are the incoming pixel color and alpha respectively. The blending result is exactly what we need for making the back-to-front composition as in Equation 2.1.

As discussed in section 2.2.2, other choices of composition include Maximum Intensity Projection and Minimum Intensity Projection. These two compositions can be done by using the library calls

```
glBlendEquationEXT(GL_FUNC_MAX_EXT)
```

or

```
glBlendEquationEXT(GL_FUNC_MIN_EXT)
```

respectively.

3.3.4 Basic Rendering Results

Figure 3.11 shows the results of the basic rendering of two bone volumes. The spine is a volume of size $128 \times 128 \times 210$ while the sacrum is of size $128 \times 128 \times 157$. The heated-object scale (section 3.4.3) is used for the color coding.

3.4 General Features

Although the major aim of VISBONE is to provide computer assistance in planning pedicle screw insertion operations, it is also expected to offer general functions for visualizing and manipulating the virtual bones. The following features aim at giving surgeons handy tools for examining a volumetric bone:

- *Clipping.* Part of the bone is removed to reveal internal structures.
- *Partitioning.* The density range is divided into several sub-ranges. User can then choose which sub-ranges are made visible.
- *Color Coding.* Different color coding schemes are applied to the volumetric data to achieve different visual effects.
- *Animations.* The visualization is animated according to the densities of different regions. This gives the user a better understanding of the spatial distribution of the BMD.
- *Measurements.* The distance between two points within the bone and the thickness of a slab cut from the bone can easily be measured.

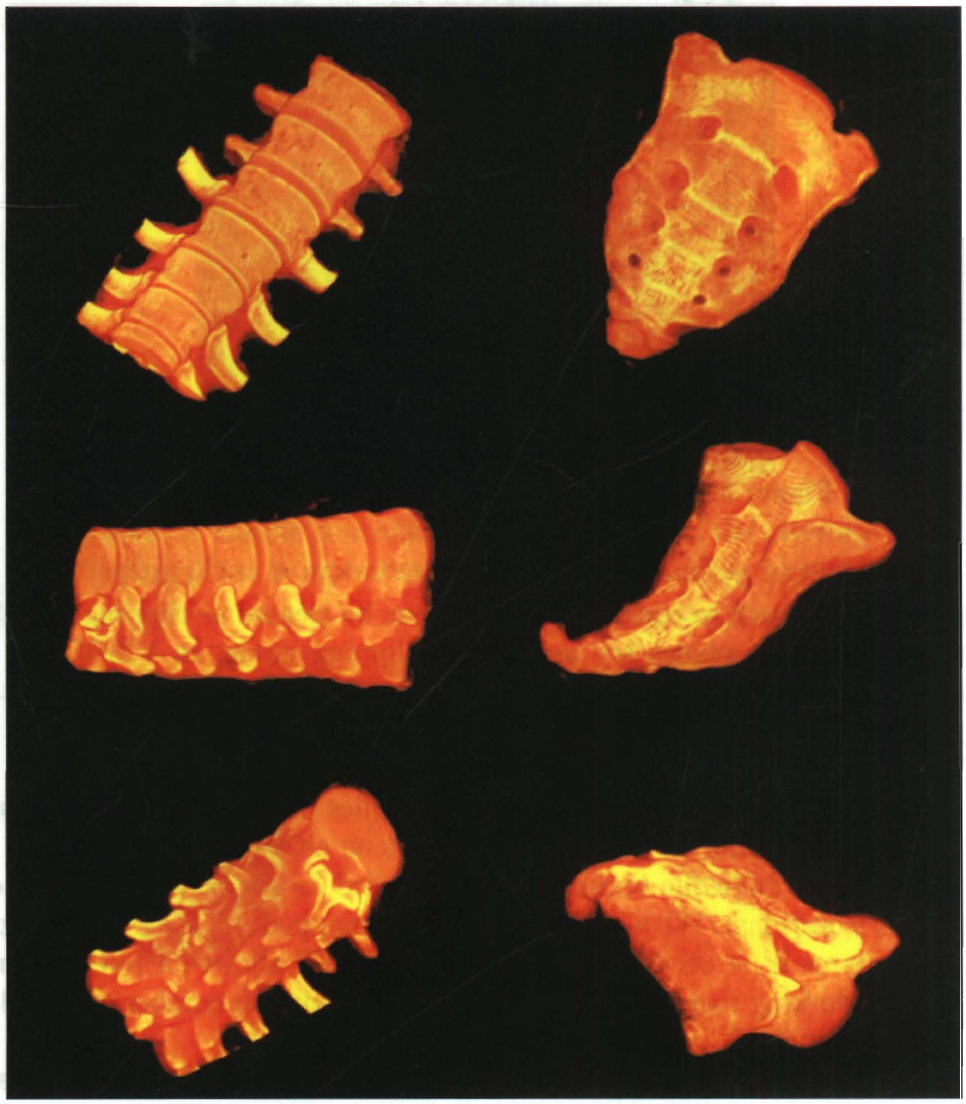


Figure 3.11: Rendering of two volumetric data sets. Left: Spine. Right: Sacrum.

- *Stereoscopic Views.* To enhance the stereopsis as perceived by the user with the help of special stereographic devices.

These features and their implementation techniques will be discussed in subsequent subsections.

3.4.1 Clipping

Clipping is provided as a basic feature in many visualization toolkits [21]. The main function of clipping in our application is to remove part of the bone to reveal the BMD distribution over the cutting plane (Figure 3.12). Although

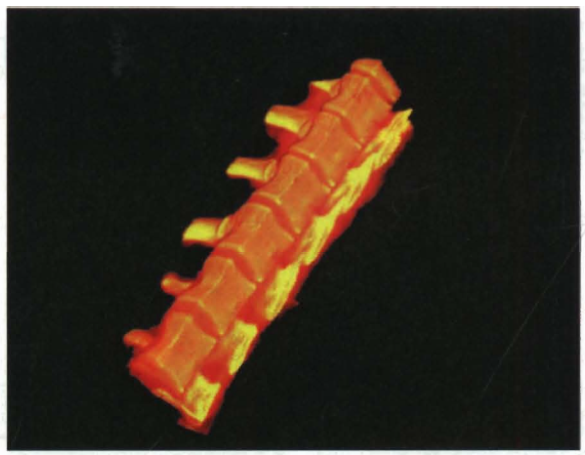


Figure 3.12: Clipping. Half of the spine is clipped.

other geometries, such as cones and spheres, can be treated as clipping primitives as described in [12], planes are used in all clipping modes in VISBONE. The reason is that surgeons are most confident when dealing with cross-sections and interpreting density changes on a planar surface.

The clipping of the volumetric data set is done by applying clip planes provided by the OpenGL functions. A clip plane divides the space into two half-space and the graphics engine only renders primitives on either side of the plane. The clip planes cut through the textured polygons used for rendering the volumes in arbitrary orientations. With the texture being mapped onto the clipped polygons, the volume is perceived as being cut. VISBONE allows for having at most 6 clip planes working together because this is the maximum number of clip planes supported by OpenGL.

Each clip plane can be arbitrarily oriented. To enhance the control, the user is provided with a handle pointing in the direction of the plane normal (Figure 3.13). The handle can be freely rotated and the orientation of the clip plane is changed. The bone volume is clipped along the plane and internal details is disclosed. An outline of the clip plane can also be drawn so that one can easily realize how the plane is oriented in the 3D space.

Block Clipping

We denote S the space where everything is defined. By using a clip plane P, we divide S into two half-space S_P^+ and S_P^-. Let us assume that contents reside in the positive half-space S_P^+ are always rendered while those reside in the negative half-space S_P^- are discarded. Notice that when there are more than one clipping planes $P_1, P_2, \ldots, P_n$, the effect is that only contents within the intersection of the positive half space of all clipping planes, i.e. $S_{P_1}^+ \cap S_{P_2}^+ \cap \cdots \cap S_{P_n}^+$, are rendered (Figure 3.14, 3.15). What if we want the union of these half-space instead? In volume rendering, there is a common visualization where a rectangular block is

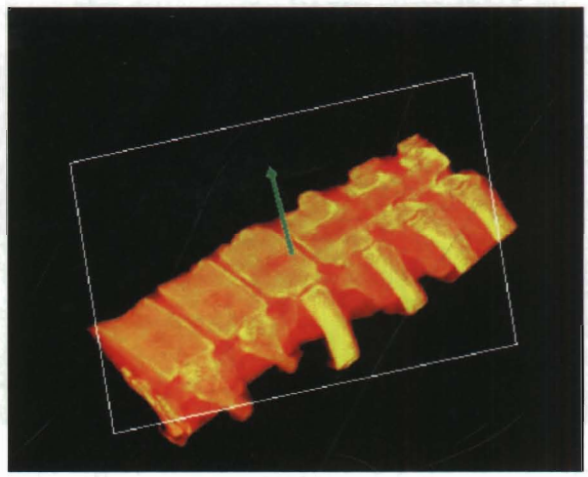


Figure 3.13: A handle controlling the orientation of the clipping plane. The clipping plane is represented by a white bounding rectangle.

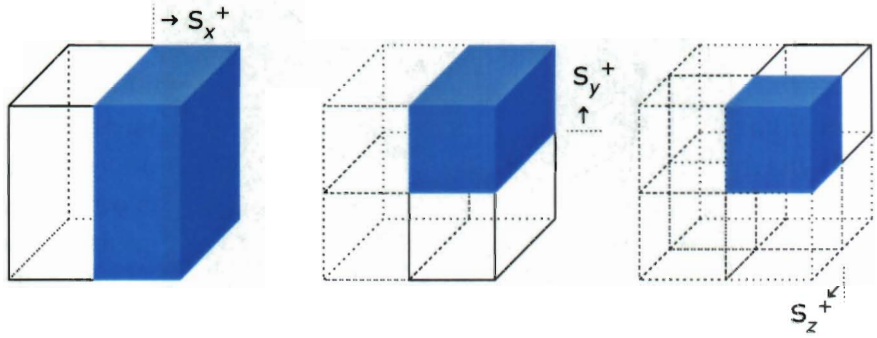


Figure 3.14: The intersection is rendered when there are more than one clip planes.

carved out and the internal details being revealed (Figure 3.16). As we shall see, the scene cannot be rendered by one pass but rather several passes, each for different parts of the volume.

Using 2D analogy as shown in Figure 3.17, the area that we want to render is given by:

$$\begin{aligned} & S_{P_1}^+ \cup S_{P_2}^+ \\ =& (S_{P_1}^+ \cup S_{P_1}^-) \cap (S_{P_1}^+ \cup S_{P_2}^+) \\ =& S_{P_1}^+ \cup (S_{P_1}^- \cap S_{P_2}^+) \end{aligned} \tag{3.1}$$

By this decomposition, we see that the required area can be rendered in two passes. The first pass uses clip plane P_1 only while the second pass uses two clip

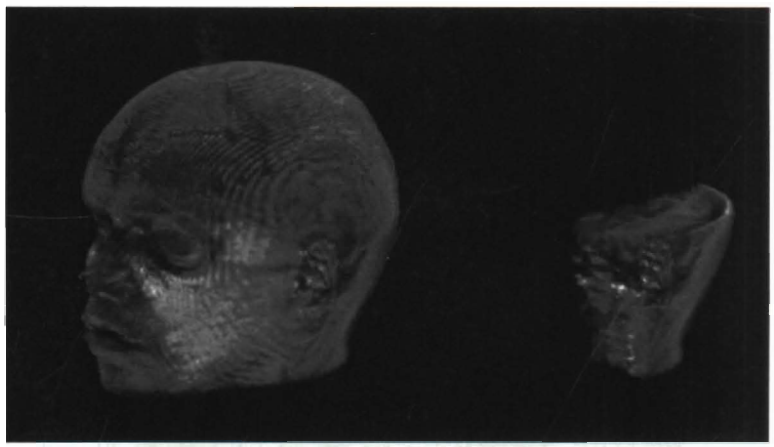


Figure 3.15: Block clipping by intersection. Left: Original Volume. Right: Applying three orthogonal clipping planes.

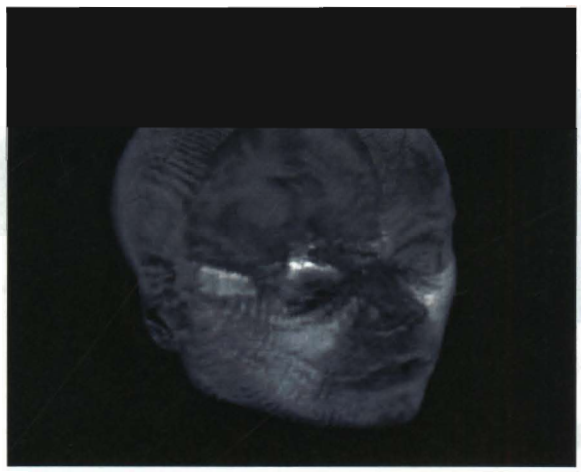


Figure 3.16: Block clipping by union. A block is calved out from the data volume.

planes, P_2 and the complement of P_1, at the same time. Hence each block is drawn only once in the multi-pass rendering.

However, when a volume is being rendered, we consider not only which blocks are drawn but also the order in which they are drawn. Since the voxels may be semi-transparent, they have to be rendered to the frame buffer in the back-to-front manner. Take the 2D case as an example again (Figure 3.17). If the viewpoint is within quadrant 3, quadrants 1 and 2 will be behind quadrant 3 with respect to the viewpoint and should be rendered first. But we can show that there is no way that quadrant 3 is drawn after both quadrants 1 and 2 if the rendering is in two passes only. If we draw $S^+_{P_1}$ first and then $S^-_{P_1} \cap S^+_{P_2}$, we

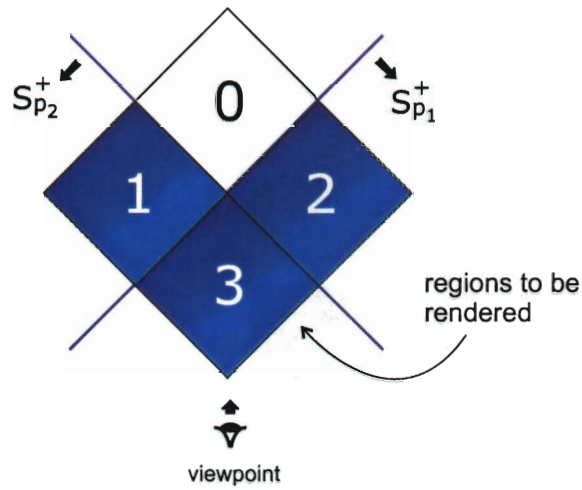


Figure 3.17: 2D analogy of block clipping by union.

shall end up with quadrant 3 drawn before quadrant 1. Or if we draw $S^+_{P_2}$ first and then $S^-_{P_2} \cap S^+_{P_1}$, quadrant 3 will be drawn before quadrant 2. In both cases, the final composition is incorrect.

Hence, we further decompose equation 3.1 so that the rendering is in 3 passes, each for drawing one single quadrant. Depending on which quadrant the current viewpoint is in, we can always find an ordering for drawing the quadrants with a correct composition. In the previous example, we choose the order "$1 \rightarrow 2 \rightarrow 3$" to accomplish the task.

In three dimensions, we have to use 7 passes for the complete rendering. We named this the *7-subcube method* since the clipped volume is divided into 7 sub-blocks. One of the possible rendering order when the viewpoint is within quadrant 0 is show in Figure 3.18.

We have also tested another method where the clipping is done by altering the shape of the textured polygons. Suppose the three clipping planes intersect at (x_0, y_0, z_0) and that the polygons are drawn parallel to the xy-plane with displacement along the z direction. We first draw rectangular polygons that cut through the data volume until we hit z_0 (Figure 3.20). This is followed by some L-shape polygons at $z > z_0$ with an inner vertex at (x_0, y_0, z). We named this the *L-shape method* which gives the same result as the 7-subcube method. The cavities of the L-shape polygons form a missing subcube, thus providing the required visualization.

We have tested the rendering time needed for different amounts of the volume being clipped away. The comparison of the frame rates in using the 7-subcube and L-shape methods is shown in Figure 3.19. In this experimental setting, the intersection of the three orthogonal clipping planes moves along the diagonal of the volume. For instance, when the intersection is 10% off the volume diagonal, most of the volume is being clipped. We can see that the two methods have very

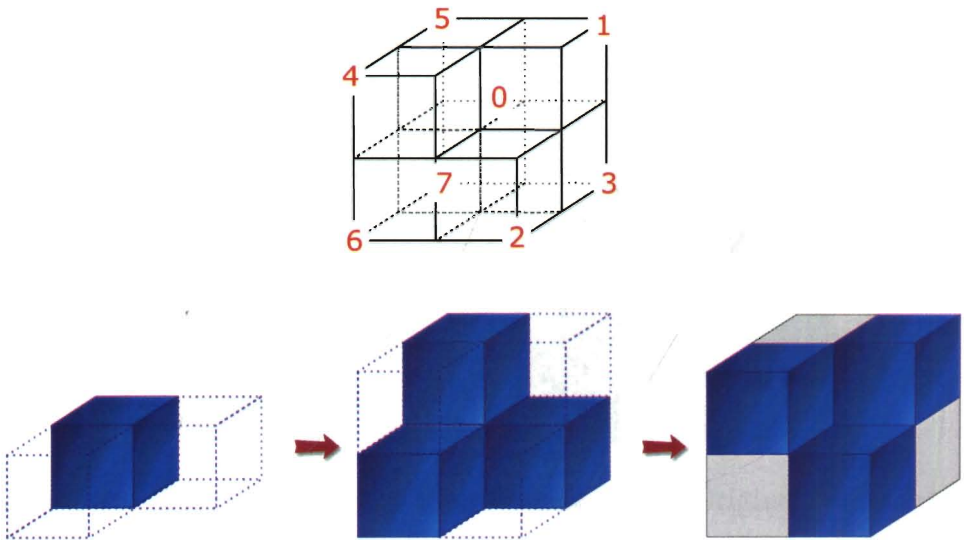


Figure 3.18: Order of rendering for 7 sub-cubes. Assuming the viewpoint is at quadrant 0, the rendering order is: $7 \rightarrow 5 \rightarrow 6 \rightarrow 3 \rightarrow 4 \rightarrow 1 \rightarrow 2$.

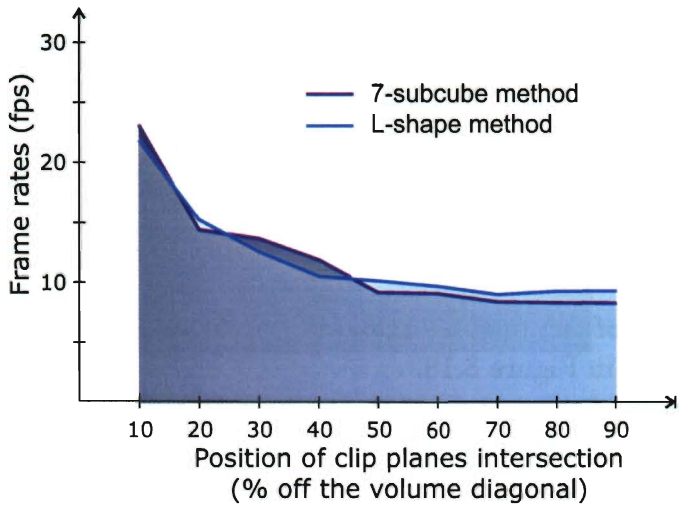


Figure 3.19: Frame rate comparison of the 7-subcube method and the L-shape method. Data set size is $128 \times 128 \times 210$.

close frame rates. The L-shape method has the advantage that we do not need to consider the rendering order of the subcubes. However, it would be difficult to determine the shapes of the polygons if the clip planes are not orthogonal to each other. In cases where the clip planes are allowed to be inclined arbitrarily to each other (but still intersected at one point), the 7-subcube method is a better choice than the L-shape method.

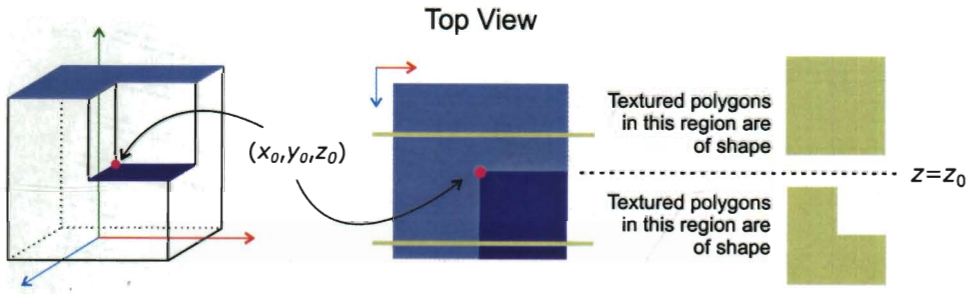


Figure 3.20: L-shape method for block clipping by changing the shape of the textured polygons.

3D Parallel Slices

A new clipping mode is tested using VISBONE. Parallel slices within the volume are drawn as shown in Figure 3.21. The slices are separated by a certain ad-

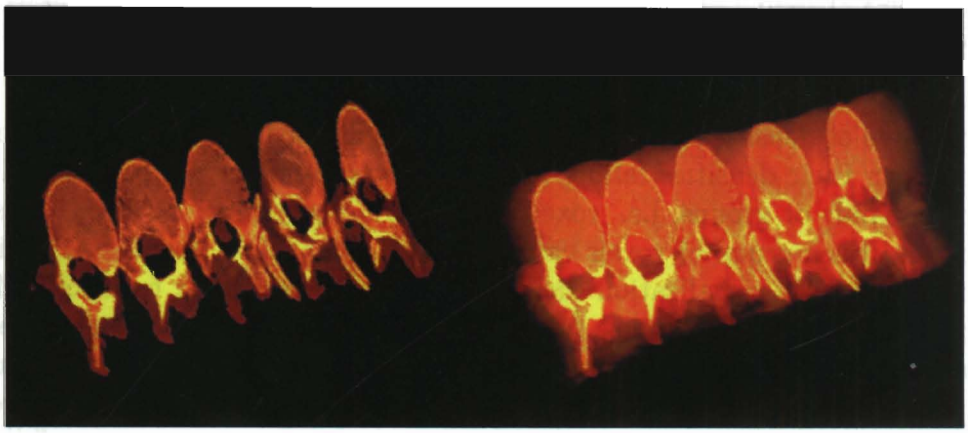


Figure 3.21: 3D Parallel slices. Left: Thin slices in 3D. Right: The rest of the volume has been made semi-transparent.

justable distance. The orientation of the slices can easily be changed by handling a normal vector to the parallel slices. Different regions are visualized when the slices slide along the plane normal. The rest of the volume is made highly transparent to provide a 3D spatial relationship between the series of 2D slices. With this setting, surgeons no longer need to rely on mental visualization to stack up the 2D CT images for a 3D picture.

Each 3D parallel slice is drawn by using two oppositely facing clipping planes (Figure 3.22). The distance between the two planes is kept to minimal so that a thin slice is shown. Series of planes are drawn in subsequent passes by displacing the clip planes for certain lateral distance. Although the entire scene is drawn in multiple passes, interactive frame rates are still maintained. This is because in

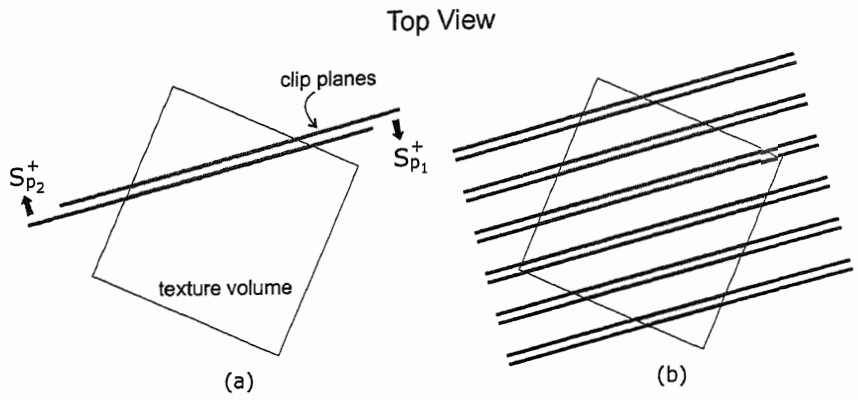


Figure 3.22: Parallel slices. (a) A thin slice formed by two clipping planes. (b) A series of parallel slices.

each pass, only a very small portion of the volume is rendered and can be done in a much shorter period of time than rendering the entire volume.

3.4.2 Partitioning

Partitioning is used to visualize the BMD distribution in different bands. The entire range of density values of the CT data is divided into several sub-ranges. Voxels with their densities falling into different sub-ranges are partitioned into different bands or groups, i.e. voxels of a group have densities within the same sub-range. Each group of voxels can be turned on or off so that they are visible or invisible. This visualization mode shows how a particular band of BMD is distributed inside the volume. Also, since most of the voxels are excluded by choosing only a band to be viewed, serious occlusion among the voxels is reduced.

Initially, boundaries of the voxel groups resulted from partitioning were expected to be as regular as mechanical parts, as by the surgeons we consulted about this feature. This turned out not to be the case because there are low density voxels scattered all over the volume. Therefore there is no obvious boundary between different bands of BMD. The reason for such discrepancy is that in their understanding surgeons have in mind a rough boundary between the bands without considering the existence of tiny pores, which are shown as low density voxels in the visualization (Figure 3.23).

Partitioning is done by changing the transfer function which maps voxel values to opacity values. Without partitioning, the opacity transfer function is a continuous function whose domain runs from the minimum voxel value to the maximum one. Now we partition the domain into 5 contiguous blocks (Figure 3.24). Each block has its opacity transfer function being the same as that of the subrange it comes from. To make one partition being invisible, the opacity transfer function of the corresponding block is set to a constant zero. This is equivalent to turning off the opacity of all voxels with densities falling into the subrange represented

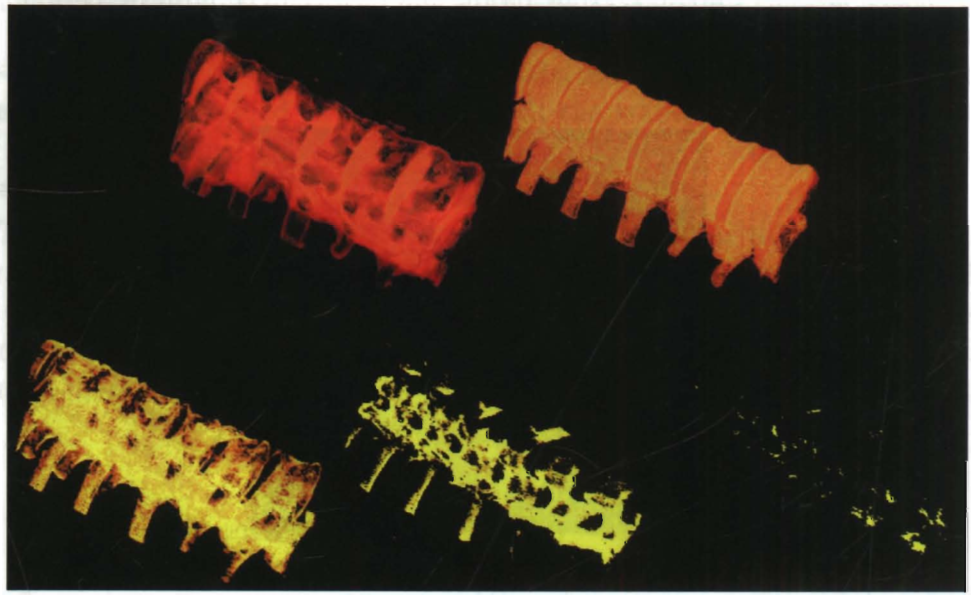


Figure 3.23: Partitioning. The density range is divided into 5 partitions. Left to right, top to bottom: Low density to high density regions.

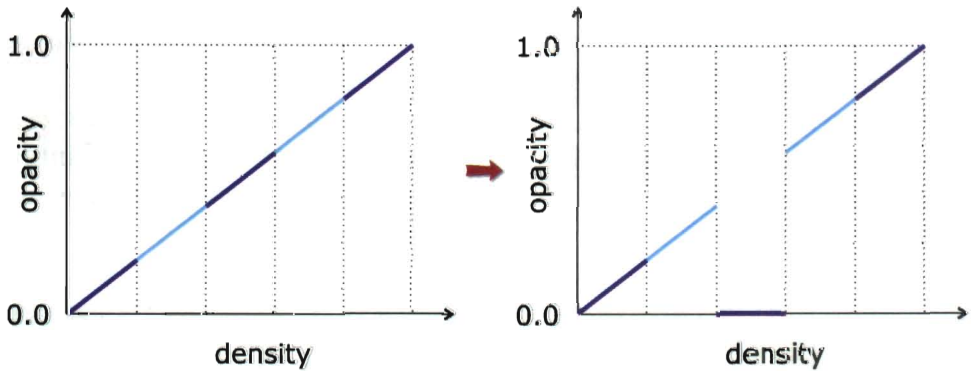


Figure 3.24: Changing the opacity transfer function to turn off the third partition.

by the partition. The partition now becomes totally transparent.

3.4.3 Color Coding

Color plays an important role in the human visual system. Voxels are assigned colors according to their BMD. Different coding schemes have different visual effects. The selection of a color coding scheme depends highly on the application. In VISBONE, several coloring schemes are available for the users to choose from. Various visual effects are achieved by combining color coding with different

opacity settings for the voxels. For instance, Figure 3.25 shows a pseudo-Xray visualization whose implementation details will be discussed later in section 4.5.1.

Figure 3.25: A pseudo X-ray visualization.

Apart from the grey scale coloring, the *heated-object scale* is also used. This color scale changes from black to white, but does not have equal red, green and blue compositions along the path like the grey scale. Instead, its red component dominates the compositions. Hence, the color of the heated-object scale is quite close to the orange hue. The color changes from brown to orange, and then becomes yellow. The underlying design principle for the heated-object scale is based on the claim that human is highly sensitive to the changes of intensities along the orange-yellow hue [17]. Figure 3.26 shows the two color scales used in VISBONE.

Figure 3.26: Color scales. Top: Grey level; Bottom: Heated-Object Scale.

3.4.4 Animation

Animation is a useful visualization technique in situations involving dynamic data transitions, such as flow visualization. Although VISBONE deals only with a static 3D scalar field, animation is still applicable. The transitions between frames are interpreted as the change of focus from one density range to another.

We use an animation to show the evolution of the bone volume from low density voxels to high density voxels. Two modes are supported: accumulative and non-accumulative. In the accumulative mode, subsequent voxels with increasing density values are added to the scene until the whole volume is visualized. In the non-accumulative mode, only voxels of a particular band are shown at any instance.

The implementation of this animation is quite similar to partitioning. However, the partition sizes are greatly reduced to allow for a smooth transition of the opacity transfer function among the animation frames. In each frame, the opacity transfer function only has a little bit difference from that of the preceding frame (Figure 3.27). Since the opacity transfer function controls the transparency

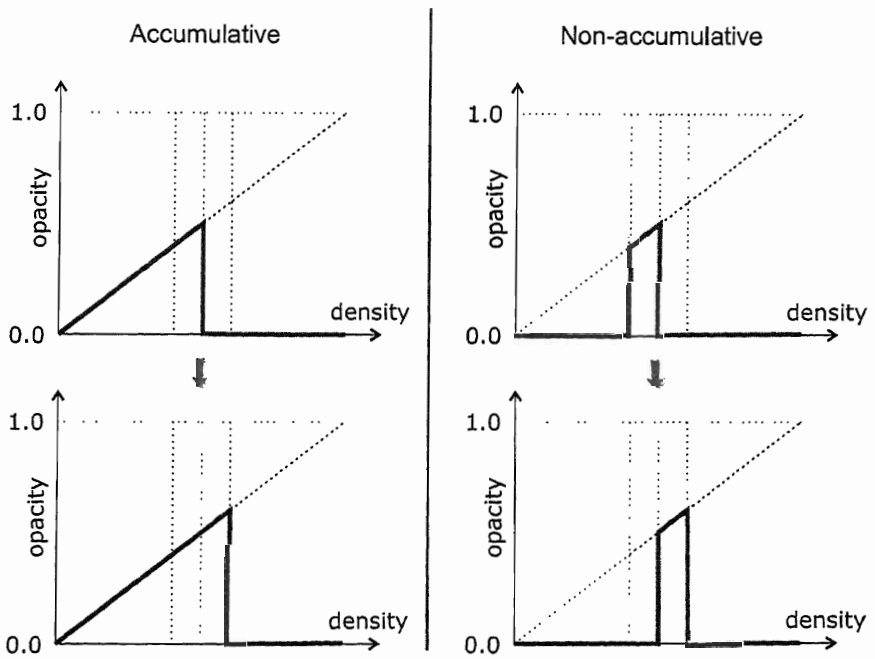


Figure 3.27: Changing the opacity transfer function for animating effects.

of each voxel, it tells the voxels how to "act" in the scene during the play. A transparent voxel can suddenly come into the scene by just changing its opacity from zero to non-zero from one frame to the next.

We say that the animation thus produced is actually an animation of the transfer function. The transfer function maps any voxel value to different opacities and colors. As the currently used bone volume encodes density information only, the animation effect is based mainly on bone densities. If each voxel also stores extra information like tissue classifications, interesting animations showing how different types of tissues evolved can be produced in the same way.

3.4.5 Measurements

Besides visualizing the BMD of the bone volume, quantitative measurements are also important for planning screw insertion. For instance, surgeons have to select screws of appropriate length for the insertion. The screw should not be too short or otherwise it will not be able to support the other implants like rods and hooks. Also, it should not be too long so as to avoid penetrating through the pedicle. Hence, two kinds of measurements, slab thickness and point-to-point distance, are provided.

Slab thickness

A slab is a sub-volume bounded by two parallel planes separated by certain distance (Figure 3.28). It has arbitrary orientation and thickness within the

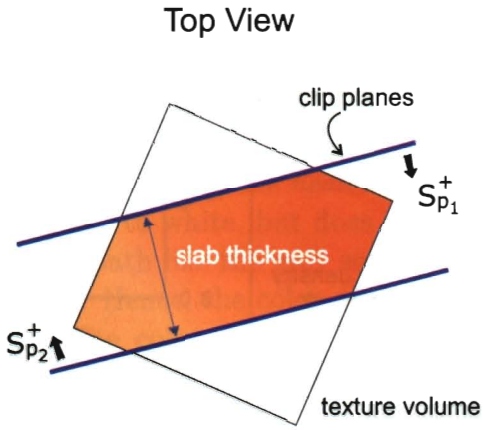


Figure 3.28: The distance between the clip planes is the slab thickness.

bone volume. A pair of parallel but oppositely facing clip planes is used to create a slab. Users can control the orientation of the planes to shape the slab. The thickness of the slab is altered by changing the distance between the two clip planes. The thickness is then derived from the separating distance of the clip planes (Figure 3.29).

Point-to-point distance

To determine an appropriate length for a screw, the best way is to measure the distance between the intended locations of the screw head and tail. This involves distance measurements in 3D space. In VISBONE, two points are picked on an appropriately positioned cutting plane (Figure 3.30). The plane is one that passes through the two points whose distance is to be measured. After specifying this guiding plane, users can move two markers freely on the plane and the distance of which is then measured.

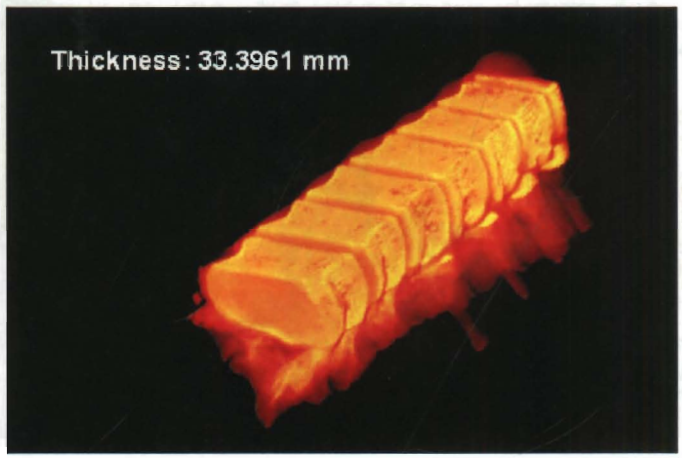


Figure 3.29: Measuring the thickness of a slab.

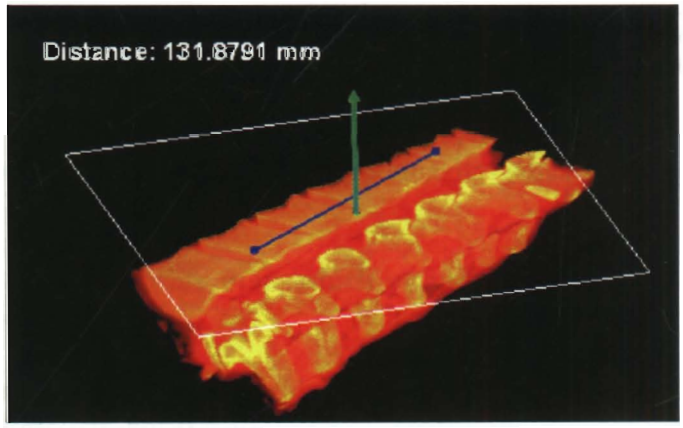


Figure 3.30: Measuring the distance between two points in a volume. The blue spots indicating two freely movable points on a plane (shown in white).

3.4.6 Stereoscopic Views

We perceive stereopsis when seeing things because we possess binocular vision. Many virtual reality applications use stereoscopic views to provide basic 3D immersive feeling. In VISBONE, stereo can be applied to the various visualizations described in previous sections. Instead of rendering the scene just once using a single projection at one viewpoint, the scene is rendered twice, once for a viewpoint corresponding to an eye position. Hence the two viewpoints are merely a slight lateral displacement of each other. With special stereographic devices like head mount display and crystal eye equipment, the two slightly different rendered images will go to each of our eyes separately (Figure 3.31). The parallax thus

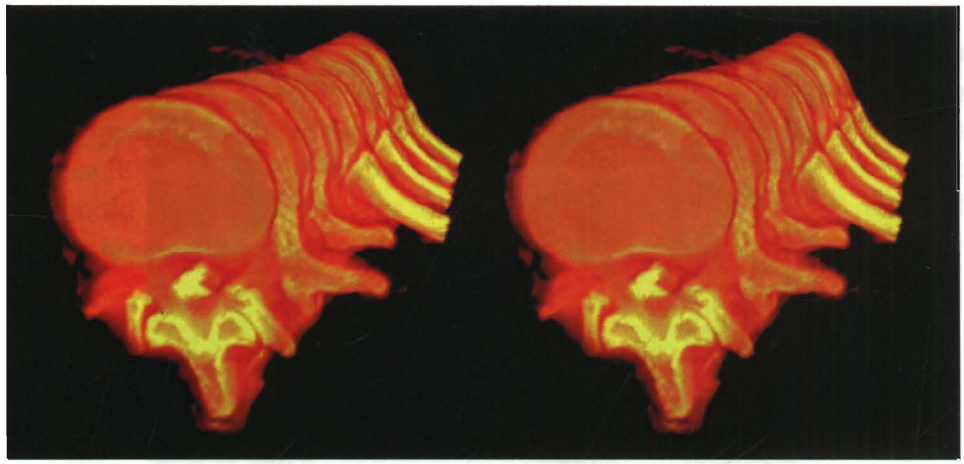


Figure 3.31: Stereo Rendering. Left, Right: The left and right views.

produced gives us a 3D illustration. However, since the scene is rendered twice for each frame, the frame rate is halved as a trade-off.

Chapter 4

Screw Insertion

Apart from the general features for 3D bone visualization and manipulation described in Chapter 3, we have also designed some special functions specifically for the requirements of assisting the screw insertion operations. These functions include:

- *Thresholding*, which enables surgeons to differentiate regions of different Bone Mineral Densities (BMD) to secure the screw;
- *Surface removal*, where a thin high density surface is removed to avoid occlusion;
- *Insertion path feasibility map*, which provides an overview for the feasibilities of insertion paths;
- *Screw insertion simulation*, which offers a real time simulation of the screw insertion process; and
- *Post-surgical evaluation*, which allows surgeons to evaluate the insertion result.

In this chapter, we shall discuss in details the techniques in accomplishing the above tasks.

4.1 Transfer Function Editing

VISBONE provides means to edit the opacity or intensity transfer functions interactively. The transfer functions are defined as piecewise functions instead of continuous ones (Figure 4.1). This offers the user more flexibilities to adjust attributes, such as color and opacity, in different density ranges. The piecewise transfer functions are independent of each other and are either linear or constant. For instance, the opacity transfer function can be made directly proportional to the density values for the low density range. It can also be inversely proportional

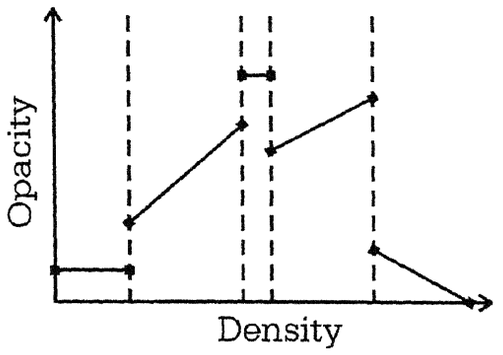


Figure 4.1: Piecewise transfer function. The function is defined independently in each density range.

to the density values for the high density range. The effect of the editing on the bone volume is visualized immediately.

Combining the piecewise transfer functions and the ability to freely divide the density range, various visual effects are achieved. With special treatments on the data volume, we may apply the effects not only according to the voxel density but also to other attributes, like location. We shall discuss how the transfer functions editing techniques are applied to the thresholding and surface removal functions.

4.2 Thresholding

When surgeons want to insert a screw into a bone, they will have to distinguish regions where it is anatomically feasible for insertion. For instance, they have to avoid nerve positions to prevent the patient suffering from neurological complications. They will then roughly divide the possible insertion regions into two classes, one with BMD that are safe for insertion while the other region has a higher risk of failure for screw insertion. A threshold BMD value is used to differentiate the two regions. Surgeons may want to view only those regions with BMD higher than the threshold so that they can examine and decide the possible orientations of the screw before the surgery. Or they may want to view regions with BMD lower than the threshold to check how vulnerable a piece of bone is.

The thresholding visualization is done by altering the opacity transfer functions. The idea is as follows: the density range is divided into two regions by a threshold value (Figure 4.2). This threshold is the lowest density value which is considered "safe" for screw insertion. By applying different transfer functions to the two regions, surgeons can control how the two regions, with BMD below and above the threshold, are displayed on the screen. To visualize only the safe regions, we set the transfer function of the lower density range to be a constant zero. The transfer function of the higher density range is one that assign opacities

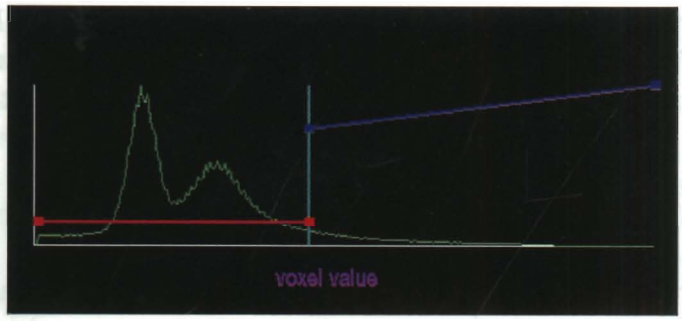


Figure 4.2: Thresholding. The green curve is the histogram. The vertical line in sky blue indicates the threshold value. The red and blue lines represent the opacity transfer functions for the regions with densities lower and higher than the threshold respectively.

directly proportional to the voxel densities. This transfer function (Figure 4.3) hides all voxels with density lower than the threshold, hence providing the required visualization. On the contrary, the transfer function shown in Figure 4.4 allows the surgeons to see only those regions with BMD lower than the threshold.

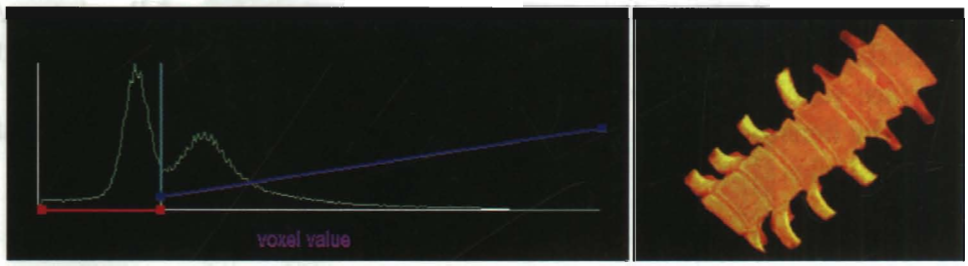


Figure 4.3: Visualizing regions with densities higher than the threshold.

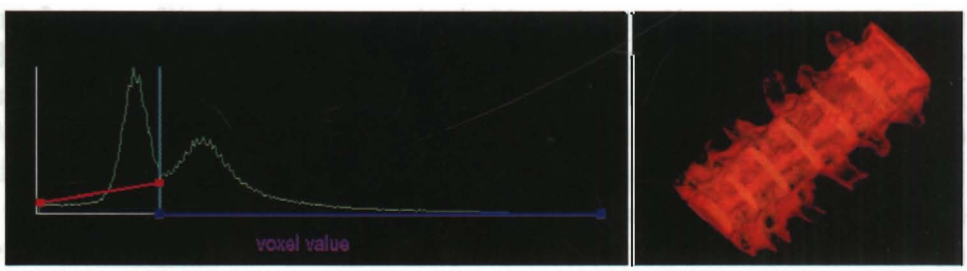


Figure 4.4: Visualizing regions with densities lower than the threshold.

4.2.1 Surface Removal

If the opacity of a voxel is made proportional to its density value, the visualization of the bone looks like a surface model. This is due to the existence of a thin layer of high density voxels lying on the bone surface. This layer makes the bone surface highly opaque, thus blocking the interior bone structures. To reveal the internal bone structures, elimination of the surface layer of the bone is mandatory. A preprocessing step is therefore needed to identify the bone surface voxels, which are then treated differently from other internal voxels.

The requirements of identifying surface voxels normally lead to a process called *segmentation* where each voxel is classified to certain category For example, a voxel can be classified according to the tissue that it is representing, e g. blood vessels, brain matters, or fat. These classifications sometimes involve computations of voxel values, not only for the voxel itself but also the surrounding voxels.

Tissue boundaries are often signified by the drastic gradient changes in neighbouring voxels. The bone surface can therefore be identified by detecting gradient changes of the voxels. However, in our application, we use a much easier approach to solve the problem. We take advantage that there is great difference in the density ranges for bone and muscle tissues. Instead of calculating the gradients for each voxel, we can easily tell whether a voxel is representing bone or muscle tissues by determining which density range has its density been fallen into. After differentiating between bone voxels and muscles voxels, the next step is to extract bone surface voxels from other bone representing voxels.

Given the density range of bone structures, a voxel is classified as bone surface voxel if it satisfies the following criteria:

1. the voxel itself is of bone density value, and

2. at least one of the 26 neighboring voxels does NOT have bone density value.

A 2D analogy is shown in Figure 4.5. The above criteria is simple and easy to test by applying a 3^3 window centred at the queried voxel. Also, the thickness of the extracted surface can readily be adjusted by altering the number of neighboring voxels being tested (Figure 4.5(b)). To obtain a surface with thickness of i voxels, the testing window should be of size $(2i+1)^3$.

After the surface voxels are identified, they are treated differently from the other voxels. The color may be changed so as to make the surface distinguishable from the others (Figure 4.6). The surface can also be made almost transparent to avoid serious occlusions.

It is clearly shown in Figure 4.7 that without surface removal, the interior bone structures cannot be seen because these regions are blocked by the high density bone surface layer. With the opacity and intensity transfer function unchanged, removing the surface layer assists in visualizing the interior tissues to some extent. In the right image of Figure 4.7, each vertebrae can be identified obviously and the interior BMD distribution is also clearly visualized.

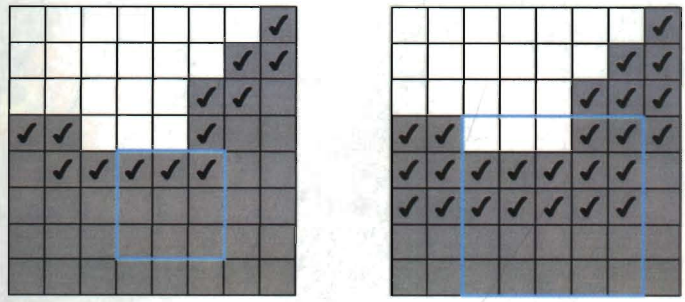


Figure 4.5: The 2D analogy of bone surface determination. (a) White and grey pixels represent soft tissue and bone, respectively. The checked pixels are classified as boundary pixels. All of them are bone pixels and at least one of the 8 neighboring pixels is a soft tissue pixel (using a 3×3 window). (b) Extracting a thicker surface with a window size of 5×5.

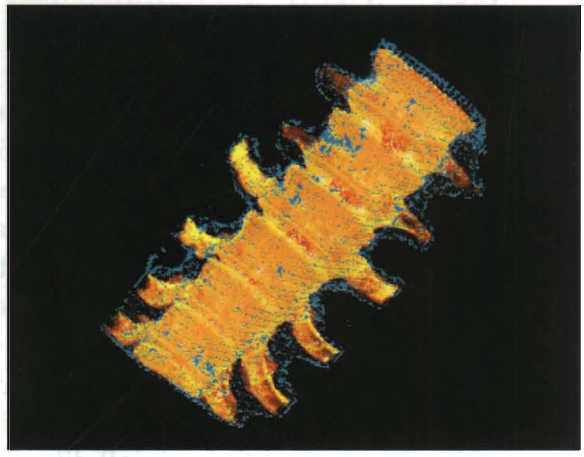


Figure 4.6: Bone with surface identified and highlighted in blue.

Surgeons sometimes would like to have a rough idea on how thick the high density surface layer is. This surface layer in general does not have uniform thickness over the bone. The difficulty in extracting the surface layer is that the inner boundary of the layer cannot be simply identified by a threshold value which differentiates the hard and soft bone tissues. The inner boundary of the surface layer depends not only on the threshold value but also on the topology and location of the surface region. It means that the threshold value may vary among different surface regions. A convenient way to tackle this problem is to peel off a surface of 1 voxel thickness, layer by layer. Surgeons can then control the number of thin layers to be peeled off until a certain portion of the volume

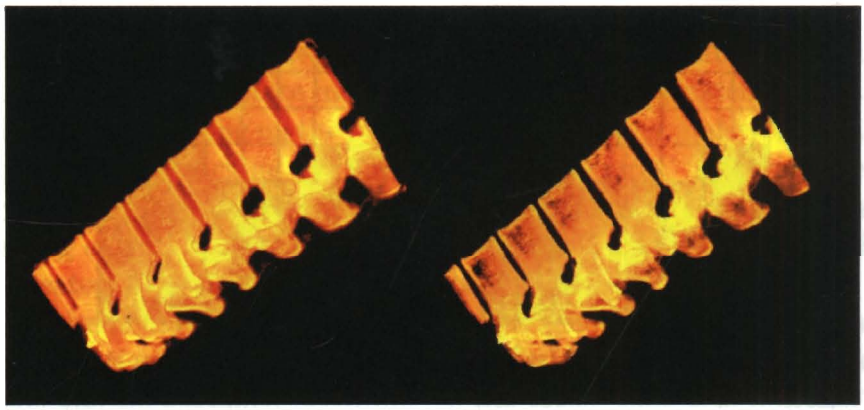


Figure 4.7: Spine with thresholding, Left: without removing the bone surface; Right: with bone surface being removed.

has been removed or a targeted region is seen.

4.3 Feasible Insertion Path

One of the surgeons' main concerns is whether a screw is inserted into bone regions with sufficient bone mineral densities. We could ask the surgeons to choose a candidate insertion path first and then the BMD information along the selected path is shown. However, when surgeons want to seek for a path with the highest BMD, they have to try all possible candidate paths and check the BMD information before they can draw a conclusion. Hence the above proposed scheme is impractical for making a fast and comprehensive decision.

In view of this, the following scheme is adopted. Once the user has specified a screw insertion point on the bone surface, VISBONE then generates a map, called "Insertion Path Feasibility Map", which indicates the feasibility of inserting the screw from the entry point into the bone volume in all directions. The minimum BMD that is encountered along each direction is shown on the map. The surgeons are hence provided with a full picture of the feasibility information on multiple insertion paths.

4.3.1 Insertion Path Feasibility Map

Map Generation

An insertion path is considered feasible if all the BMD along the path are above a preset threshold. This gives us a convenient interpretation because we can immediately tell whether a path is feasible by determining the minimum BMD along that path. If the minimum BMD is higher than the threshold value, the path is said to be feasible; otherwise, it is cast as an infeasible one.

As discussed before, the task of telling whether a single path is feasible is relatively easy. But what if we want to know about the feasibilities of all possible paths at the same time? To tackle this problem, we first notice that given the insertion point, all possible insertion paths will pass through the point and are similar to a family of rays originated from it. To find the minimum BMD along a path is analogous to have a minimum intensity projection along a ray in the same direction and orientation as the path. Hence, to qualify all paths originated from an insertion point at the same time, we perform a perspective minimum intensity projection (MIP) with the insertion point acting as the ray converging point. The ray-casting approach is applied by casting rays from the insertion point to an image plane positioned near to it. Each pixel on the image plane represents a particular path orientation. By selecting the minimum density along each ray, a map showing the feasibility of all insertion paths in one single image is generated. We called this an "Insertion Path Feasibility Map" (Figure 4.8). In fact, because

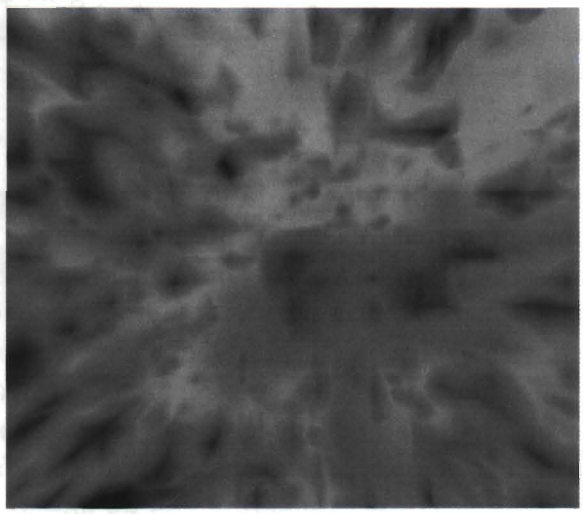


Figure 4.8: An Insertion Path Feasibility Map. Brighter area represents screw insertion orientations that are more feasible.

of this special setting, only the feasibility of insertion paths having intersections with the image plane can be shown. However, this is good enough to serve our purposes because an insertion point is always located on the bone surface. In general, possible path orientations cover no more than a half-space defined by the image plane. Moreover, surgeons can change the view direction to capture the feasibilities of more insertion paths.

There is also an important difference between the ordinary ray-casting approach and the one that we used here in generating the map. For ordinary ray-casting, a ray is cast indefinitely until no voxel that affects the composition is hit by the ray. However, our intention is to check only whether a screw is securely inserted into the bone. There is no need to consider regions where the screw cannot reach. This means that the rays should travel as far as the screw

can reach but stop thereafter. The region covered by the screw is in fact a spherical volume with its center located at the insertion point, and its radius being the length of the insertion. Hence, the length of the rays should be limited to be the same as the radius. This is mandatory for filtering out voxels that the screw cannot reach to prevent these voxels from affecting the final result.

The ray-casting approach involves intensive computations and is difficult to be done with interactive frame rate. To achieve fast and responsive perspective MIP, we make use of the OpenGL `GL_MIN_EXT` blending equation for a Minimum Intensity Projection composition (refer to section 2.2.2). The viewpoint is located at the insertion point so that the composition is along rays originated from it. An image plane is placed very close to the viewpoint to capture the feasibility information of intercepted paths. All pixels in the frame buffer must be first initialized with the maximum intensity. Textured polygons are then drawn and the minimum intensity will remain in the frame buffer for each pixel.

We also rely on the stencil buffer to limit the effective rendering region to be the spherical subvolume centered at the insertion point. The sphere should have the same size as the one mentioned above, i.e. its radius is equal to the insertion length. The sphere is also used as a clipping geometry. The method for volume rendering with arbitrary clipping geometry described in [12] has been adapted and modified. The region within the sphere is rendered, instead of being clipped away. The original method suggests that two passes of rendering be done, one for back-facing fragments and another for front-facing fragments as described in section 2.2.3 for correct rendering. As we have described, there may be extra back-facing fragments that have been rendered during the first pass but do not belong to the intersection between the textured polygon and the sphere (Figure 4.9(a)). The front-facing fragments are rendered in the second pass to recover these erroneous regions thus produced.

However, we found in our adaptation of the method that if the viewpoint is placed at the center of the sphere, there is no need for the second pass at all. This is because if we are looking from the inside of the sphere, the back-facing fragments drawn must fall within the intersecting region (Figure 4.9(b)). In other words, the stencil buffer is set up correctly in the first pass. This has waived the need for the two-pass rendering. The steps for setting up the stencil buffer for rendering a spherical subvolume centered at the viewpoint are shown as follows:

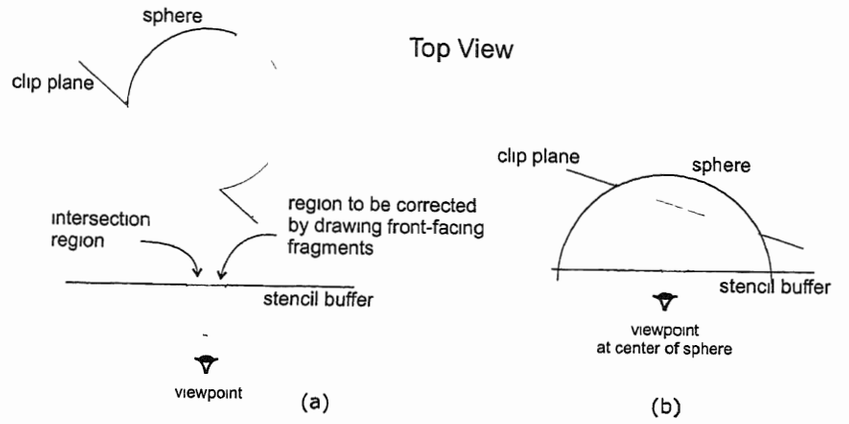


Figure 4.9: Stencil buffer setting. (a) With viewpoint outside of the sphere; (b) With viewpoint at the center of the sphere.

For each textured polygon to be drawn,

1. Fill the entire background of the viewport with maximum intensity
2. Clear the stencil buffer with value 0
3. Disable writing to the frame buffer
4. Set up a clip plane of the same orientation and position as the textured polygon
5. Set up stencil function for stencil testing,
   a) using 1 bit testing, and only write to the stencil buffer where the value is 0:
      `glStencilFunc(GL_EQUAL, 0x0, 0x1);`
   b) invert the stencil buffer value when fragments are written:
      `glStencilOp(GL_KEEP, GL_KEEP, GL_INVERT);`
6. Disable write to the depth buffer
7. Draw a sphere under the effect of the clip plane
8. Enable write to the frame buffer
9. Set up stencil function for stencil testing,
   a) using 1 bit testing, and only write to the stencil buffer where the value has been set:
      `glStencilFunc(GL_NOTEQUAL, 0x0, 0x1);`
   b) do not modify the stencil buffer during the process:
      `glStencilOp(GL_KEEP, GL_KEEP, GL_KEEP);`
10. Draw the textured polygon

Figure 4.10 shows the result of hardware accelerated perspective MIP. The image highly resembles that of the relatively slow perspective MIP by ray-casting while attaining real-time rendering.

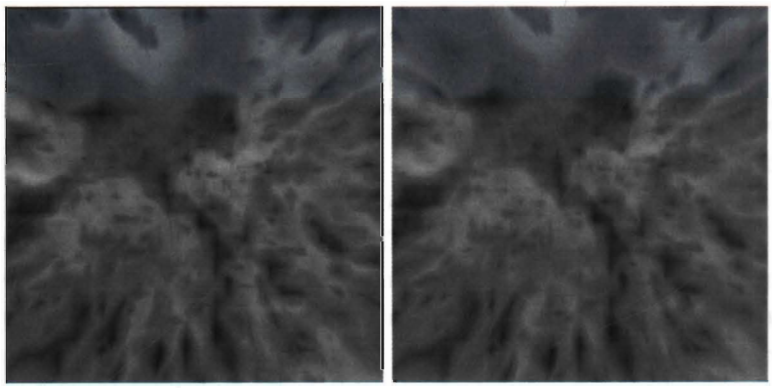


Figure 4.10: Comparisons of the quality of feasibility maps generated by, Left: ray-casting MIP; Right: hardware accelerated MIP.

Pores Removal

Although perspective MIP provides means to obtain information on insertion path feasibility, it is vulnerable to situations where an unqualified voxel of low density is very close to the insertion point. The voxel affects more paths than when it is farther away (Figure 4.11). All paths passing through it are claimed

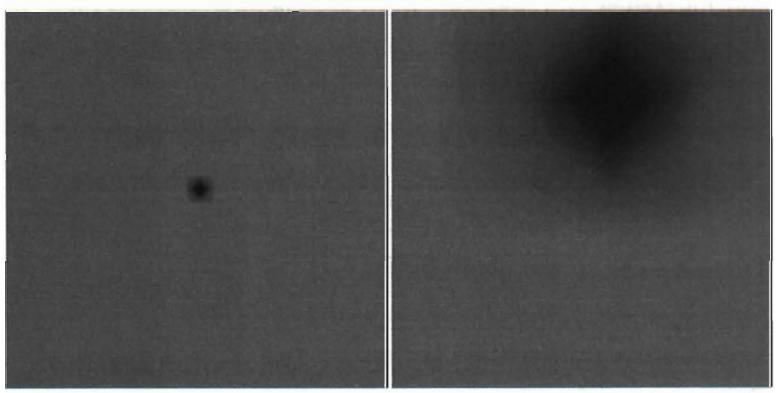


Figure 4.11: Perspective MIP of a low density voxel, Left: far away from; Right: close to, the screw insertion point.

unqualified. On the other hand, the definition of a feasible path may as well be relieved to allow for the existence of small pores along the path. Unqualified

regions consisting of voxels with density lower than the threshold may still contain feasible paths if the regions are not significantly large.

To address the above issues, the bone volume is preprocessed to remove small pores with densities lower than the preset threshold. A common image processing technique called *closing* is applied to the volume but in a 3D manner. A 3D filter is set up whose dimension is half of the maximum dimension allowed for an unqualified region. After the closing operation, low density regions smaller than the maximum dimension allowed are filled up.

The closing operation is decomposed into two stages: dilation followed by erosion. The dilation filter is applied to qualified voxels and fills up unqualified voxels within the filter. In order to ensure the same result regardless of the sequence in applying the filter to the voxels, an unqualified voxel will take the maximum density of its surrounding qualified voxels. This also eliminates the influence of the voxel in the MIP process.

The algorithm for applying the dilation filter is as follows:

Let D be the dilation-filtered volume and V be the original volume.
Let $V(i)$ denote the BMD of voxel i in V.
$D \leftarrow V$
    For each voxel i with $V(i) > threshold$
        center a $d \times d \times d$ filter O at i, where d is
        half the maximum dimension of unqualified region
        For each voxel j in O
            If $V(j) < threshold$ and $D(j) < V(i)$
            then $D(j) \leftarrow V(i)$

Figure 4.12(a) and (b) show the perspective MIP of the original volume and the result after applying the dilation filter. Small unqualified regions are filled up. However, the sizes of large unqualified regions are also reduced.

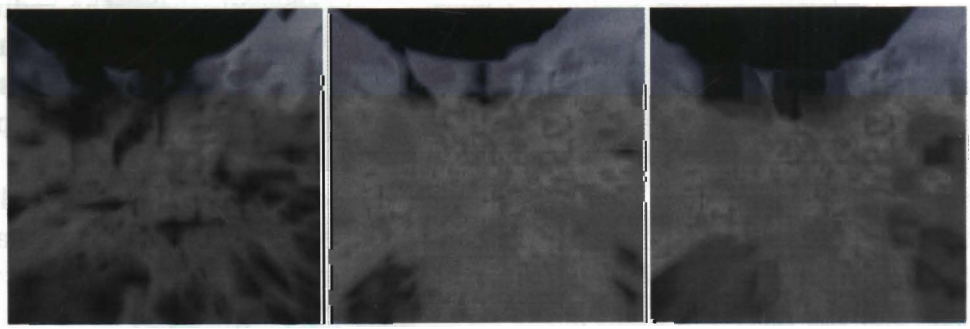


Figure 4.12: The closing operation. (a) Original perspective MIP; (b) After dilation; (c) After erosion.

As part of the closing operation, the second step is to apply the erosion filter to moderate the result. The erosion filter is applied to unqualified voxels and enlarges unqualified regions. Since small unqualified regions are filled up by the dilation filter, erosion has no effect in these regions. The algorithm for applying the erosion filter is as follows:

```
Let E be the erosion-filtered volume and D be the dilation-filtered volume.
Let D(i) denote the BMD of voxel i in D.
E ← D
    For each voxel i with D(i) < threshold
        center a d × d × d filter O at D(i), where d is
        half the maximum dimension of unqualified region
        For each voxel j in O
            If E(j) > threshold and E(j) < D(i)
            then E(j) ← D(i))
```

Figure 4.12(c) shows the result of applying the erosion filter. Note the recovery of the large unqualified region (in dark) at the top of the image.

After the closing operation, the filtered volume is then used for producing the Insertion Path Feasibility Map. False reports on infeasible paths that are caused by small unqualified regions are eliminated.

4.3.2 Screw Orientation to Map Correspondence

We have shown how VISBONE tells the feasibility of all possible insertion paths directed from a given insertion point on the bone surface. The information of feasibilities is contained on a planar Insertion Path Feasibility Map. We have to provide means for the surgeons to make correspondences between each pixel on the map and the associating insertion path orientation. This is done in VISBONE by giving two views. The first control view offers a 3D bone visualization with a spherical subvolume centered at the screw insertion point being carved out using the arbitrary geometry clipping method described in [12]. The radius of the sphere is equal to the length of the insertion as selected by the user. A pointer representing the insertion direction has its tail being fixed at the screw insertion point and the head being able to move freely on the spherical clipping boundary (Figure 4.13). At the same time, the Insertion Path Feasibility Map appears in the second view side-by-side with the control view (Figure 4.14). A hollow disc is drawn on the map. We can imagine there is an invisible handle connecting the insertion point and the center of the disc. The distance between the disc and the insertion point is the screw insertion length. The normal of the plane containing the disc always points towards the insertion point. The invisible handle experiences the same orientation as the pointer which is controlled by the user in the first view. As the controlling pointer moves, the circle on the second view encloses the region on the map and gives the feasibility information for

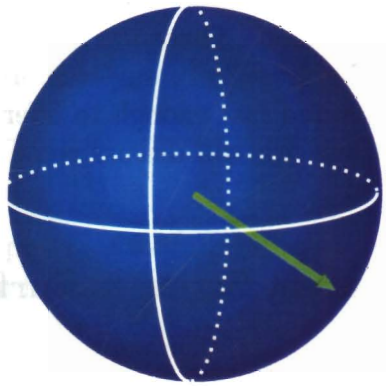


Figure 4.13: A handle with its head moving freely on a spherical surface. The pointer corresponds to a screw insertion direction.

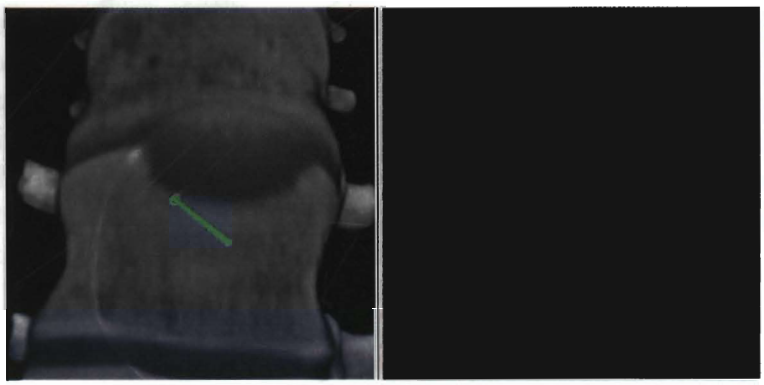


Figure 4.14: Map correspondence. Left: A spherical subvolume is removed from the bone and a pointer represents the insertion orientation. Right: A marker indicates the region corresponding to the direction of the pointer.

the path orientation that the controlling pointer is directed at. Correspondences between 3D orientations and the 2D map are thus established.

4.4 Screw Insertion Simulation

Sometimes the surgeons not only want to know where a screw should be inserted into the bone, but also how the screw is to be inserted. A real-time simulation of the bone insertion process is thus incorporated. By this simulation, surgeons can preview the surgical operation. For example, they can get an idea on the spatial relationship between the bone and the screw before insertion, where the screw should be inserted, and the orientation of the screw during the insertion, all in three dimensions. This kind of simulation provides visual experience to surgeons so that they can discuss the operation beforehand. They will then have a better understanding and planning for the operation.

To make a screw insertion simulation possible, we have to show a screw moving freely within the bone volume. There are several concerns that we have to consider. How to model the screw? In surfaces or in volumes? Is it opaque or not? Is the rendering method fast enough to offer an interactive simulation? In the subsequent subsections, we will discuss some possible solutions to these problems.

4.4.1 Semi-Transparent Screw in Surface Representation

Screws are normally opaque objects. The reason why we want to consider them as semi-transparent objects is because we would like to explore whether semi-transparent surgical tools can provide special visual assistance. Hence the first possible measure we considered is to have a semi-transparent screw with a surface representation being inserted into a bone volume.

When we first approach this task, we immediately face the problems of how to decide the correct rendering order for both the screw and the bone. This can be transformed into a problem of rendering a scene of semi-transparent polygons placed all over the space. Remember that there is an explicit rendering order for the textured polygons. Now we want to insert into this ordering some more semi-transparent polygons representing the screw surface. There might be even intersections between the screw polygons with the textured polygons for the bone. If this situation arises, we have to subdivide the screw polygons along the textured polygons until a correct rendering order is resolved. However, in our simulation of screw insertion, the position of the screw changes from time to time, which means that the ordering needs to be updated from frame to frame. We believe that an interactive simulation can hardly be possible in view of the work that has to be done in arranging the rendering order. As a result, we move on to consider the next solution.

4.4.2 Opaque Screw in Surface Representation

The next solution that we considered for the task of having a screw that moves freely inside a bone is to have an opaque screw in surface representation. As the screw is opaque, the rendering order can be arranged easily. This is done by

drawing the screw first, and then the textured polygons in the original sequence. One thing we have to be careful is that bone regions occluded by the opaque screw should not be rendered, and this is ensured by proper setting of the depth buffer. Basically, the rendering procedure is as follows:

To render an opaque screw in surface representation within a bone volume:

1. Clear the depth buffer
2. Enable write to depth buffer and enable depth test
3. Draw the screw
4. Draw textured polygons for the bone

Although this method provides us with interactive rendering, it does not give the necessary visualization in certain circumstances. For instance, we may see a hollow screw if the bone volume is clipped (Figure 4.15). Hence, our next target

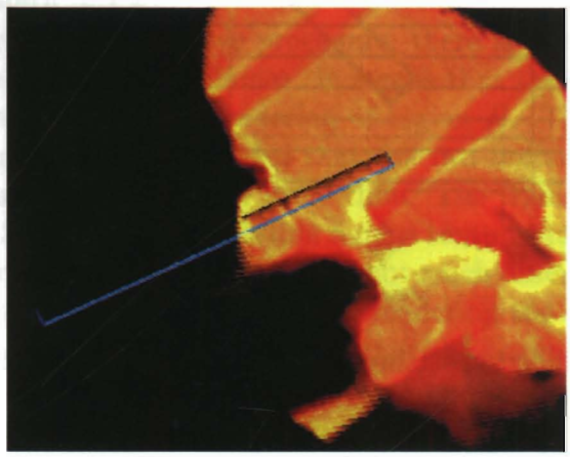


Figure 4.15: An opaque surface-based screw inserted into the bone volume. A hollow screw interior is revealed under the effect of a clip plane.

is to model the screw as volumetric data instead.

4.4.3 Semi-Transparent Screw in Volume Representation

One way to otain a volumetric representation of a screw is to scan the screw using a CT device. The CT data thus acquired will then constitute the required

volume. There are also research studies in converting surfaces into volume representations [18]. We may also used the proposed algorithms to convert some CAD models of the screw into volumes.

Once we have a screw volume at hand, we can treat it as an ordinary volumetric data set. It means that we can draw a set of parallel textured polygons to render the screw. Now, we have two sets of textured polygons, one for the screw and one for the bone. Since the screw is semi-transparent and resides within the bone volume, we have to consider the rendering order of the two sets of polygons. To simplify the task, we fix the orientation of the textured polygons for the screw to be the same as that for the bone. In this setting, the textured polygons for the screw are no longer texture axis aligned but cut through the screw volume at arbitrary angles (Figure 4.16). As explained in section 3.3.1, this amounts

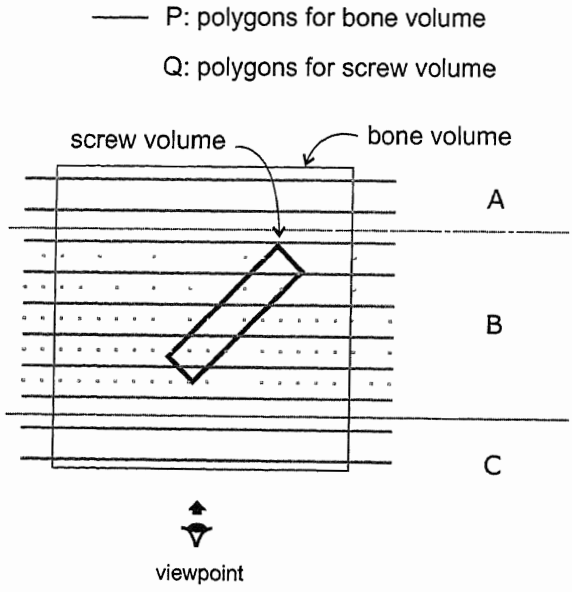


Figure 4.16: Rendering setting up of a semi-transparent screw in volume representation moving in the bone volume.

to a slower rendering than using texture axis aligned polygons because of the difference in tri-linear interpolation calculations. However, the screw volume is usually of a relatively small size when compared with the bone. For example, the volume dimension of a screw is $32 \times 32 \times 128$ while the size of the spine is as large as $128 \times 128 \times 256$. Therefore the slight increase in rendering time for the screw by not using texture aligned polygons is insignificant in the total rendering time.

The steps for visualizing a semi-transparent screw in volume representation that moves within a bone volume is as follows:

Let P and Q be the two sets of textured polygons for the bone and the screw respectively
Assume the screw volume totally resides in the bone volume. We then divide the rendering region into three parts: A, B, C as shown in Figure 4.16, and:

- A is the region where there are only polygons from P and is the region farthest away from the viewpoint;
- B is the region where there are polygons both from P and Q; and
- C is the region where there are only polygons from P and is the region closest to the viewpoint.

To render the entire scene:

1. draw polygons from P falling within region A in back-to-front manner
2. sort the polygons from P and Q falling within region B in decreasing distances from the viewpoint; and draw these polygons in back-to-front manner
3. draw polygons from P falling within region C in back-to-front manner

There are other cases where the screw volume does not reside totally in the bone volume as shown in Figure 4.16 and regions are defined similarly.

In step 2 described in the above procedure, there is interleaved rendering of polygons from the two texture volumes. It means that the two textures are referenced alternatively during the rendering of region B. There is no significant degradation of rendering time provided that the two textures can be accommodated in the texture memory at the same time. The frame rate of rendering a screw (of size $32 \times 32 \times 128$) moving within a bone texture (of size $128 \times 128 \times 128$) is around 9 to 10 frames per second. There is not much difference when compared with the frame rates shown in Figure 3.7 for rendering the bone volume alone.

However, if the two textures are so large that they cannot be accommodated entirely within the texture memory, memory swapping must take place. This increases the rendering time drastically for each frame. In our implementation, one voxel takes up 4 bytes when represented by four components in RGBA. Hence, a $128 \times 128 \times 256$ texture volume occupies 16 Mbytes which is the maximum size offered by the texture memory of our testing hardware. In this case, no matter how small the screw is, swapping of the two textures in the texture memory is inevitable. Using a screw size of $32 \times 32 \times 128$, the I/O overhead thus induced degrades the performance to a frame rate of around 0.2 fps which is absolutely unacceptable.

To avoid the performance degradation due to texture memory swapping, we should make sure that the two texture volumes can be accommodated in the texture memory simultaneously. One may consider reducing the texture size of the bone by resampling. However, this will amount to loss of bone details to some extent.

4.4.4 Opaque Screw in Volume Representation

If we do not want to reduce the size and hence the resolution of the bone volume, we may limit ourselves to consider opaque screw only. The performance concern for the previous method is because of the need for drawing interleaving textured polygons from two distinct textures, which in turn is due solely to the semi-transparent nature of the screw. If the screw is opaque instead, interleaving polygons are no longer necessary, and there is only one texture swapping operation for drawing each frame. In this case, the screw volume is loaded first for rendering the whole set of textured polygons for the screw, and then the bone volume is loaded for drawing the textured polygons for the bone.

Here, we have to be careful about the depth values being written to the depth buffer. When a textured polygon is drawn, the depth values of the polygons are recorded in the depth buffer. However, we only want to record the depth values of those fragments that represent the screw but not empty space. This is to make sure that in the subsequent rendering of the textured polygons for the bone, only those regions not occluded by the screw are rendered.

The alpha testing has been used to limit the drawing of the polygons to regions of the screw only. Since the screw is opaque, writing to the frame buffer and the depth buffer is limited to those fragments with opacity greater than 0.5. The rendering is thus controlled in the specified manner.

Here is the rendering procedure:

To render an opaque screw in volume representation within a bone volume:

1. Clear the depth buffer
2. Enable write to depth buffer and enable depth test
3. Enable alpha testing to allow fragments with opacity > 0.5 only
4. Disable depth test and alpha test
5. Draw the screw
6. Draw textured polygons for the bone

Steps 5 and 6 in the above procedure show that two textures can be loaded into the texture memory one by one. Hence this method can be considered as an alternative to the previous method when the resources, i.e. texture memory, are limited. The only thing we need to sacrifice is the semi-transparent property of the screw. The two methods are implemented in VISBONE as supplement to each other. The rendering result is shown in Figure 4.17.

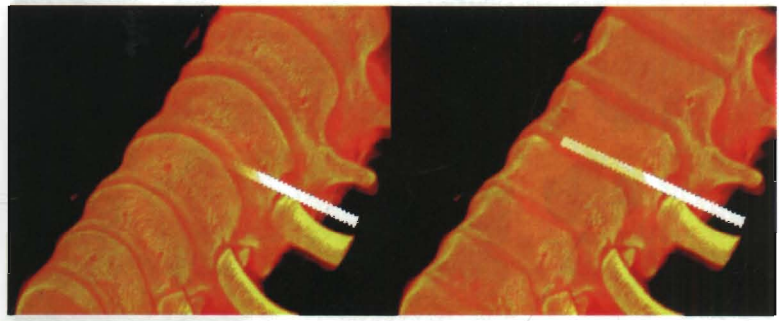


Figure 4.17: Opaque Screw in Volume Representation. Right: Part of the bone being clipped to show screw penetration.

4.5 Post-Surgical Evaluation

By using the Feasibility Path Insertion Map and the screw insertion simulation, surgeons can determine the best choice for screw insertion. After the insertion simulation, surgeons also want to have means to evaluate the resulting implanted structures. They may use the general features provided by VISBONE as described in section 3.4 for examining the implanted bone in 3D. However, we believe that surgeons will feel more comfortable and confident if they are provided with tools similar to their current practice. Hence, the following two special features dedicated to post-surgical evaluation are included.

4.5.1 Pseudo-Xray Imaging

X-ray is the most commonly used images for presenting bone implants. It provides an overview of the structures affected by the surgical operation when compared with CT which only offers sectional information. As described in section 2.3.1, the X-ray image is formed due to the accumulated attenuation of penetrating rays. The harder the tissue, the brighter the projected regions on the image. To approximate the X-ray effect in VISBONE, we add up the intensity of corresponding voxels along a ray and the following OpenGL blend functions are used:

```
glBlendEquationEXT(GL_FUNC_ADD_EXT);
glBlendFunc(GL_SRC_ALPHA, GL_ONE);
```

We sometimes have an "over-exposed" image as a result of the pseudo-Xray imaging (Figure 4.18). This happens when the sum of the intensities of the voxels along a ray exceeds the maximum intensity. It is the same as actual X-ray imaging where radiologists adjust the strength of the penetrating rays to produce images with different contrasts. Unlike radiologists, we alter the intensities of the voxels with the bone volume, so as to scale down the accumulated intensities. However, we do not need to change the voxel density directly. Instead, we scale down

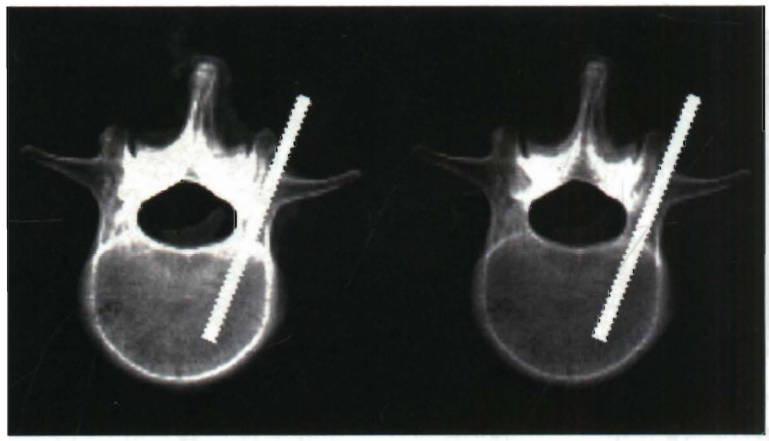


Figure 4.18: Pseudo-Xray. Left: An "over-exposed" image. Right: A moderated image by scaling down the voxel opacities.

the assigned opacities of the voxels by making changes on the opacity transfer function. In this way, we can produce pseudo-Xray images with the best contrasts for evaluation.

4.5.2 Sectional Slices Generation

Another tool of high utilization by the surgeons is the sectional CT slices. Consider this as the reverse process of the data acquisition stage in VISBONE where CT slices are stacked together to form a bone volume. Now we want to obtain these slices back from the bone volume with the screw being inserted. These slices are no longer restricted in certain orientation but are arbitrarily oriented. If the two volumes are merged together, we can easily get a slice of it by drawing a rectangle representing the required image in the texture space with appropriate texture coordinates attached to the vertices of the polygon (Figure 4.19(a)). The merging of the two volumes requires putting the screw volume in a proper position and replacing voxels in the bone occupied by the screw with the screw density.

If we want to generate the sectional slices directly from the visualization described in section 4.4.3 and 4.4.4, a special treatment is necessary. To generate an image for a sectional slice, a pair of clip planes is set up after the orientation of the slices is selected. The two clip planes are parallel and opposite to each other and are in the selected orientation. They clip away the rest of the volume while only a thin sectional slice remains. However, the thin slice is in general not parallel to the image plane. Hence, the next thing we need to do is to set up an orthogonal projection along the normal of the sectional slice (Figure 4.19(b)). A series of images are then generated by moving the clipping planes along the plane normal. Figure 4.20 shows a series of images of a screw being inserted into the spine.

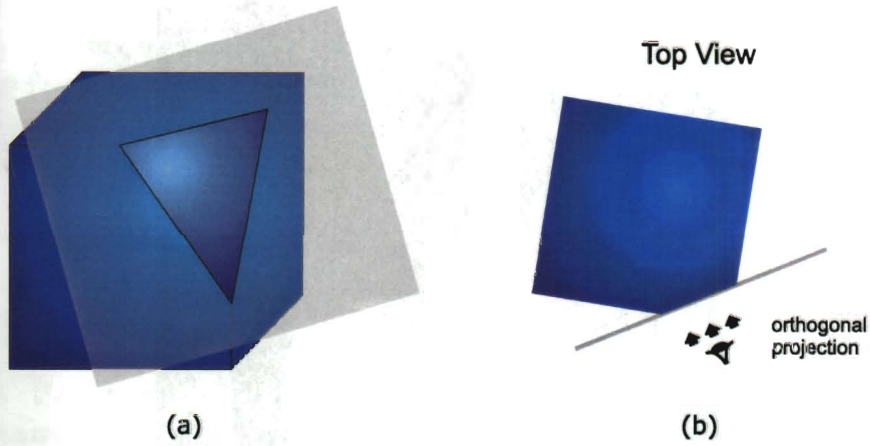


Figure 4.19: Sectional slices generation settings. (a) Using a textured polygon at arbitrary orientation for a merged texture volume of the bone and the screw. (b) Using a pair of clipping planes and an orthogonal projection when the two texture volumes are not merged.

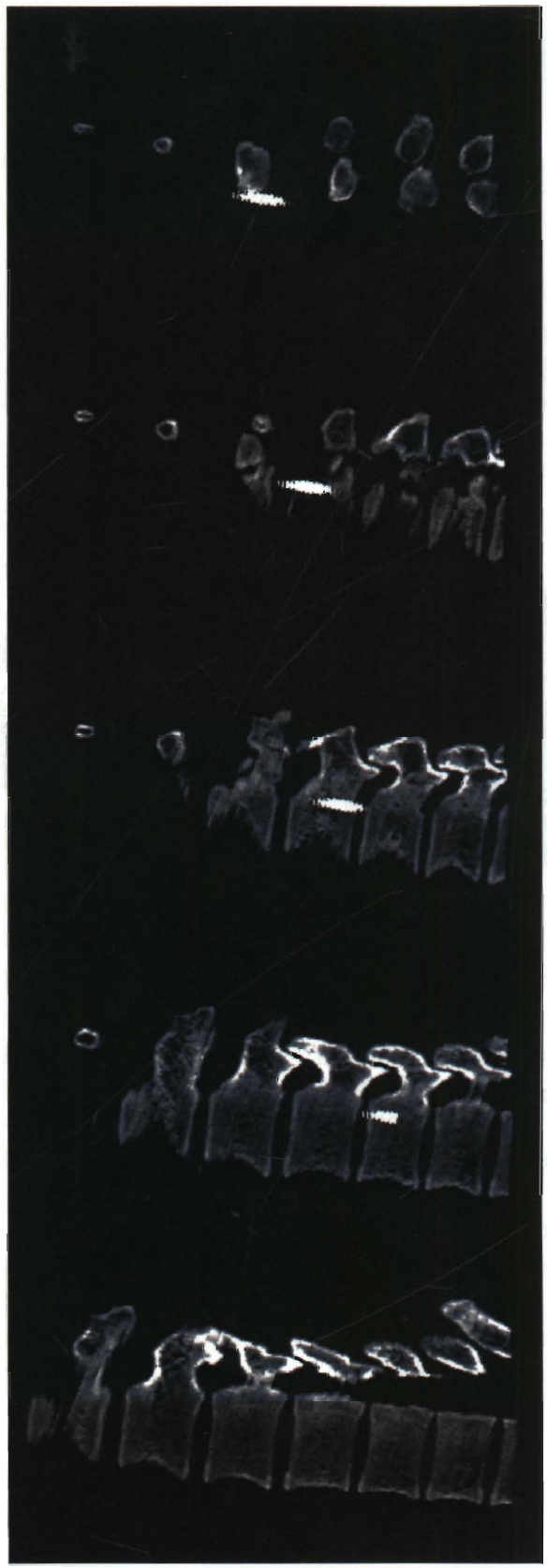


Figure 4.20: A series of CT sectional slices at arbitrary orientation showing a screw being inserted into the spine.

Chapter 5

Conclusions

In this thesis, we have presented a pilot visualization system called *VISBONE*, which is intended to assist surgeons in analyzing Bone Mineral Density distribution for planning pedicle screw insertion. The Bone Mineral Density distribution is a vital decisive element for determining screw insertion positions during an implant operation. The success of this surgery highly depends on whether the screw is inserted to an appropriate region whose densities are sufficient to "secure" the screw.

A visualization system capable of assisting surgeons to make related surgical decisions in the context of Bone Mineral Density is most welcomed. It has been observed that the special requirements raised by this application cannot be fulfilled by the common features available in most general-purpose visualization toolkits. Hence, we need to develop a new volume-based visualization system with a number of tailor-made functions for our particular application.

Various computer visualization techniques have been incorporated into VISBONE. Among the features supported by VISBONE, some are of general purposes while the others are designed specifically for the aspect of surgical planning for pedicle screw insertion. The general-purpose features include 3D bone interactive examination, different kinds of clipping at any arbitrary orientation, distance and slab thickness measurements, BMD thresholding and partitioning, color coding, animations, and stereoscopic supports.

As far as surgical planning is concerned, the visualization system allows a simulation of screw insertion at any point on the bone surface. Once the insertion location is fixed, the system provides the BMD information along all possible screw orientations. We introduce a new hardware supported method in generating the "Feasibility Path Insertion Map". Surgeons may then decide which is the best direction for inserting the implants based on the information provided by the map. They can also monitor the screw insertion process by a real-time simulation. To make a complete surgical planning, we also provide handy tools such as pseudo-Xray and sectional slices at arbitrary orientation for the surgeons to evaluate the pre-surgical results.

Throughout the thesis, we show the feasibility in applying computer visualization techniques in surgical planning for pedicle screw insertion. This thesis,

however, does not signify the end of the VISBONE project, but rather it denotes a new stage of turning the experimental system into a clinical application. VISBONE is developed on Onyx, a high-end SGI workstation which is capable of providing 3D accelerated texture mapping. The prohibitive cost of Onyx is hardly affordable by any ordinary orthopaedic clinics. Fortunately, we now have a special hardware chipset called *VolumePro*[1] available in the market of reasonable cost. It is a graphics processor which supports volume rendering in real time by the latest hardware technology in handling volumetric data sets. Most important of all, it is designed specifically for PC platforms. It follows naturally that our next step is to migrate VISBONE so that it can work on PC with the help of VolumePro. There is also a need to integrate VISBONE with the other existing clinical applications such as the CT imaging system for smooth data transitions and operations.

Although VISBONE is intended for surgical operation planning, it can easily be extended for teaching and training purposes. With special haptic devices, students will experience a virtual surgical operation and learn the techniques of implants insertion. Instead of practising initial routine work on expensive high-quality synthetic bone models, students can now reinforce the techniques, tools and implants by using the virtual system.

The entire system will be subjected to clinical evaluation once the PC version is ready. It will be made available to surgeons, teaching staff and medical students. The system will be further reviewed based on the collected comments and suggestions. The significance of computer visualization assistance in the surgical planning process is determined. A mature system will be put into clinical usage afterwards. All these efforts aim to provide direct benefits to the patients of related surgeries.

Throughout the entire study, we found that we are not merely providing computerized assistance to surgeons, but in return, we are also inspired by the specific problems or issues that arise from the application. As a concluding remark, we sincerely hope that this work will open up a new vision for both the orthopaedics and computer graphics specialists. Possibilities of further collaborations between the two disciplines will be explored.

[1] `http://www.rtviz.com/`

Bibliography

[1] F.A. Joelsz, R. Kikinis, and F. Shtern. The Vision of Image-Guided Computerized Surgery: The High-Tech Operating Room. *Computer-Integrated Surgery: Technology and Clinical Applications*, R.H. Taylor et. al., MIT Press, 1996.

[2] D. Schlenzka, T. Laine. Computer-Assisted Pedicle Screw Insertion – First Clinical Experience. *Computer Assisted Orthopedic Surgery (CAOS)*. Hogrefe & Huber Publishers, Seattle·Toronto·Bern·Göttingen, 1999, pp. 99-103.

[3] K. Kumano, S. Hirabayashi, Y. Ogawa, Y. Aota. Pedicle Screws and Bone Mineral Density. *Spine*, Volume 19, Number 10, 1994, pp. 1157-1161.

[4] T. Laine, D. Schlenzka, K. Mäkitalo, K. Tallroth, L. Nolte, and H. Visarius. Improved Accuracy of Pedicle Screw Insertion With Computer-Assisted Surgery. *Spine*, Volume 22, Number 11, 1997, pp. 1254-1258.

[5] W.E. Lorensen and H.E. Cline. Marching Cubes: A High Resolution 3D Surface Construction Algorithm.

[6] M. Levoy. Display of surfaces from volume data. *IEEE Computer Graphics and Applications*, 8(3):29-37, March 1988.

[7] D. Laur and P. Hanrahan. Hierarchical Splatting: A Progressive Refinement Algorithm for Volume Rendering. *ACM Computer Graphics, Proc. SIGGRAPH '91*, pp. 285-288.

[8] Y.Y. Shin, S. Napel, and G.D. Rubin. Fast Sliding Thin Slab Volume Visualization. *ACM Symposium on Volume Visualization '96*, pp. 79-86.

[9] H. Pfister, A. Kaufman, and F. Wessels. Towards a Scalable Architecture for Real-Time Volume Rendering. *Proceedings of the 1995 Eurographics Workshop on Graphics Hardware*, pp. 123-130, Maastricht, The Netherlands, August 1995.

[10] B. Cabral, N. Cam, and J. Foran. Accelerated Volume Rendering and Tomographic Reconstruction Using Texture Mapping Hardware. *ACM Symposium on Volume Visualization '94*, pp. 91-98.

[11] A. Van Gelder and K. Kwansik. Direct Volume Rendering with Shading via Three-Dimensional Textures. *ACM Symposium on Volume Visualization '96*, pp. 23-30.

[12] R. Westermann and T. Ertl. Efficiently Using Graphics Hardware in Volume Rendering Applications. *ACM Computer Graphics, Proc. SIGGRAPH '98*, pp.169-177.

[13] C. Paggetti, P. Dario, T. Ciucci, B. Allotta, and M. Marcacci. A Computer-Based System for Assistance to Surgery. *Computer Assisted Orthopedic Surgery (CAOS)*. Hogrefe & Huber Publishers, Seattle·Toronto·Bern·Göttingen, 1999, pp. 69-79.

[14] L.J. Brewster, S.S. Trivedi, H.K. Tuy, and J.K. Udupa. Interactive Surgical Planning. *IEEE Computer Graphics and Applications*, Volume 4, Number 3, March 1984, pp. 31-40.

[15] M.W. Vannier, J.L. Marsh, and J.O. Warren. Three-Dimensional Computer Graphics for Craniofacial Surgical Planning and Evaluation. *Computer Graphics*, Volume 17, Number 3, July 1983, pp. 263-273.

[16] J.C. Russ. *The Image Processing Handbook*, 2nd ed. CRC Press, 1995.

[17] H. Levkowitz. *Color Theory and Modeling for Computer Graphics, Visualization, and Multimedia Applications*, Kluwer Academic Publishers, 1997.

[18] A. Kaufman. Efficient Algorithms for 3D Scan-Conversion of Parametric Curves, Surfaces, and Volumes. *Computer Graphics*, Volume 21, Number 4, July 1987, pp. 171-179.

[19] K. Akeley. Reality Engine Graphics. *ACM Computer Graphics, Proc. SIGGRAPH '93*, pp. 109-116, July 1993.

[20] P. Haeberli and M. Segal. Texture Mapping as a Fundamental Drawing Primitive. *Proc. Fourth Eurographics Workshop on Rendering 1993*, pp. 259-166.

[21] W. Schroeder, K. Martin and, B. Lorensen. *The Visualization Toolkit*, 2^{nd} ed., Prentice Hall, 1998.

[22] A. Kaufman. *Volume Visualization.* IEEE Computer Society Press, Los Alamitos, CA, 1991.

[23] B. Lichtenbelt, R. Crane, and S. Naqvi. *Introduction to Volume Rendering.* Hewlett-Packard Company, 1998.

[24] M. LaRouere, J. Niparko, S. Gebarski, and J. Kemink. Three-dimensional X-ray Computed Tomography of the Temporal Bone as an Aid to Surgical Planning. *Otalaryngology-Head and Neck Surgery*, Volume 103, Number 5, Part 1, November 1990, pp. 740-747.

[25] M. Kamimura, S. Ebara, H. Itoh, Y. Tateiwa T. Kinoshita and K. Takaoka. Accurate Pedicle Screw Insertion Under the Control of a Computer Assisted Image Guiding System: Laboratory Test and Clinical Study. *Journal of Orthopaedic Science*, Number 4, 1999, pp. 197-206.

[26] P. Sprawls, Jr. *Physical Principles of Medical Imaging*, 2nd ed., Medical Physics Pub., 1995.

[27] T. Doby and G. Alker. *Origins and Development of Medical Imaging*, Southern Illinois University Press, 1997.

[28] Z.H. Cho, J.P. Jones and M. Singh. *Foundations of Medical Imaging*, Wiley, 1993.

[29] D.R. Ney, E.K. Fishman, D. Magid and R.A. Drebin. Volumetric Rendering of Computed Tomography Data: Principles and Techniques. *IEEE Computer Graphics & Applications*, Volume 9, Number 2, pp. 24-32, March 1990.