High-resolution CT data acquisition software and 3D visualization tool

Peng Fan
Iowa State University

Follow this and additional works at: https://lib.dr.iastate.edu/rtd

Recommended Citation
Fan, Peng, "High-resolution CT data acquisition software and 3D visualization tool" (2001). Retrospective Theses and Dissertations. 21195.
https://lib.dr.iastate.edu/rtd/21195

This Thesis is brought to you for free and open access by the Iowa State University Capstones, Theses and Dissertations at Iowa State University Digital Repository. It has been accepted for inclusion in Retrospective Theses and Dissertations by an authorized administrator of Iowa State University Digital Repository. For more information, please contact digirep@iastate.edu.
High-resolution CT data acquisition software and 3D visualization tool

by

Peng Fan

A thesis submitted to the graduate faculty
in partial fulfillment of the requirements for the degree of
MASTER OF SCIENCE

Major: Computer Engineering
Major Professors: Doug Jacobson and Joseph N. Gray

Iowa State University
Ames, Iowa
2001
This is to certify that the Master's thesis of

Peng Fan

has met the thesis requirements of Iowa State University

Signatures have been redacted for privacy
# TABLE OF CONTENTS

LIST OF FIGURES .............................................................................................................. vi
LIST OF TABLES .............................................................................................................. viii
ACKNOWLEDGEMENTS .................................................................................................. ix
ABSTRACT ............................................................................................................................ x

## CHAPTER 1. INTRODUCTION ......................................................................................... 1

1.1 Computed tomography ............................................................................................... 1
1.2 Data visualization ....................................................................................................... 2
1.3 Motivation and goal of this research .......................................................................... 3
1.4 Chapter overview ....................................................................................................... 4

## CHAPTER 2. VOLUME RENDERING METHOD ................................................................. 5

2.1 Basic concept: Pixels and Voxels .............................................................................. 5
2.2 Perspective and orthographic projections ................................................................... 5
2.3 Volume rendering pipeline ....................................................................................... 6
2.4 Volume rendering methods ...................................................................................... 8
   2.4.1 Coordinate and transform matrix ....................................................................... 8
   2.4.2 Forward / inverse projection and transformation .............................................. 10
   2.4.3 Splatting ........................................................................................................... 11
   2.4.4 Ray Cast Inverse Mapping rendering ............................................................... 12
   2.4.5 Shading ............................................................................................................ 14
      2.4.5.1 Phong illumination model ........................................................................... 14
      2.4.5.2 Shading in Volume Rendering ................................................................. 16
2.5 Comparison between object order rendering and image order rendering .................. 18

## CHAPTER 3. 3D CT VISUALIZATION TOOL ................................................................. 19

3.1 Introduction ................................................................................................................. 19
3.2 Software development .............................................................................................. 20
   3.2.1 Software architecture ....................................................................................... 20
   3.2.2 C3D class ......................................................................................................... 21
3.2.2.1 Data members ................................................................. 21
3.2.2.2 Function members .......................................................... 22
3.2.2.3 Rendering volume function ............................................. 24
3.3 CT volume data manipulation .............................................. 25
3.3.1 Import and output data ..................................................... 25
3.3.2 Data display controls ......................................................... 27
   3.3.2.1 Histogram based look up and transparency ................. 27
   3.3.2.2 Normalization ............................................................... 31
   3.3.2.3 Clipping and ROI .......................................................... 33
   3.3.2.4 Animation ................................................................. 35
3.4 Performance evaluation ....................................................... 36

CHAPTER 4. HIGH RESOLUTION CT SCAN SOFTWARE .................. 38
4.1 Introduction ........................................................................... 38
4.2 System description ............................................................. 39
   4.2.1 Digital X-ray imaging subsystem PaxScan 2520 .............. 39
   4.2.2 Software layer for digital X-ray imaging subsystem .......... 41
   4.2.3 Software layer for image acquisition from frame grabber .... 44
   4.2.4 Motion control indexer (AT6000) ................................. 45
   4.2.5 The overall structure of the software ......................... 46
4.3 Implementation of the software ......................................... 46
   4.3.1 Acquisition X-ray image .............................................. 46
   4.3.2 Xray image manipulation .............................................. 47
   4.3.3 Motion control ............................................................... 49
   4.3.4 CT scan ................................................................. 51
   4.3.5 Automatic scan ............................................................. 53
   4.3.6 System calibration .......................................................... 54
4.4 System demonstration ..................................................... 56

CHAPTER 5. CONCLUSIONS AND FUTURE WORK .................. 62
5.1 Conclusions ................................................................. 62
5.2 Future work ................................................................. 63
LIST OF FIGURES

Figure 2.1 Voxels and volume ................................................................. 5
Figure 2.2 Perspective projection .......................................................... 6
Figure 2.3 Orthographic projection ....................................................... 6
Figure 2.4 Volume rendering pipeline .................................................. 7
Figure 2.5 Specify the original position of the 3D data set ..................... 9
Figure 2.6 Ray cast inverse mapping ..................................................... 12
Figure 2.7 Ray cast inverse mapping procedure .................................... 14
Figure 2.8 Diffuse reflection ................................................................. 15
Figure 2.9 Specular reflection ............................................................... 16
Figure 3.1 A volume with regular profile ............................................. 19
Figure 3.2 Software structure for 3D volume rendering ....................... 20
Figure 3.3 Reduce the step range along the ray .................................... 23
Figure 3.4 Open ASCII file dialog ....................................................... 26
Figure 3.5 View setting page ............................................................... 26
Figure 3.6 Progress bar ..................................................................... 27
Figure 3.7 Intensity page ................................................................. 27
Figure 3.8 (a) Rendering mass soil ....................................................... 28
Figure 3.8 (b) Display the porosity of soil sample ............................... 29
Figure 3.9 Mapping page ............................................................... 30
Figure 3.10 Rendering image with different mapping range ................. 30
Figure 3.11 Normalization slices (a) Before normalization (b) After normalization ....... 31
Figure 3.12 Normalization page ............................................................ 32
Figure 3.13 Spatial range page .......................................................... 33
Figure 3.14 ROI clip page ............................................................... 34
Figure 3.15 Padding (a) ROI Clipping without Padding (b) ROI Clipping with Padding ... 34
Figure 3.16 Animation dialog .......................................................... 35
Figure 3.17 A sequence JPG images for animation ............................. 36
Figure 4.1 Digitization with CCD array camera ................................................................. 38
Figure 4.2 CT data acquisition ............................................................................................ 39
Figure 4.3 Sinogram of a cross section with rotation step 1° ............................................. 39
Figure 4.4 Internal configuration of the receptor ............................................................... 40
Figure 4.5 Software layers of digital X-ray imaging subsystem .................................... 42
Figure 4.6 Software layers of image acquisition from frame grabber ............................. 44
Figure 4.7 Software layers of AT6000 indexer ................................................................. 45
Figure 4.8 Image acquisition page .................................................................................... 47
Figure 4.9 (a) X-ray image (b) Histogram (c) Slice ........................................................... 48
Figure 4.10 Two sorts of implement methods for scaling functions ................................. 49
Figure 4.11 Directions of motion control ....................................................................... 50
Figure 4.12 Motion control page ...................................................................................... 50
Figure 4.13 (a) Setup image (b) CT scan setting page ..................................................... 51
Figure 4.14 CT control page ......................................................................................... 52
Figure 4.15 Automated Scan Setting ............................................................................... 53
Figure 4.16 (a) System calibration page (b) Image of dead pixels ................................. 54
Figure 4.17 Central of Rotation ....................................................................................... 55
Figure 4.18 Setup for the CT scan .................................................................................. 56
Figure 4.19 Scan setting for test ..................................................................................... 57
Figure 4.20 Sinogram of 10 cross sections ..................................................................... 58
Figure 4.21 Reconstructed cross sectional image from sinogram ................................ 58
Figure 4.22 Rendered image of the whole CT data set .................................................. 59
Figure 4.23 (a) Rendering the entire coupon (b) Cut the coupon at y=120 .................... 60
Figure 4.24 Visualizing the porosity of the part ............................................................... 61
LIST OF TABLES

Table 4.1 Table 4-1 Image Acquisition Modes ................................................................. 41
ACKNOWLEDGEMENTS

My sincere thanks to Dr. Joe Gray for his countless help and valuable suggestions on the thesis. Without his guidance and support, this research work wouldn’t be possible. Thank you Jiadong, Terry, Jianing, Junhong for your supports during my research. I would also like to thank Dr. Doug Jacobson and Dr. Satish Udpa for their valuable time and help they provided throughout this work.
ABSTRACT

Computer tomography (CT) technology which allows cross sectional imaging of objects without disassembling them plays an important role in diagnostic medicine and industry nondestructive evaluation. Modern CT data acquisition systems can collect a large amount of CT data to construct the interior structure as well as the external surface of an object. 3D visualization is an important tool to understand and analyze these huge data. In this thesis, control software for a high resolution X-ray data acquisition system and a 3D visualization tool are developed.

The high resolution X-ray scan system of the X-ray group at Center of Nondestructive Evaluation consists of motion control, X-ray source, detector and frame grabber. The new amorphous silicon detector is used to replace the image intensifier and CCD camera and provides high resolution X-ray images. The control software developed in this thesis integrates image acquisition, CT scan, automatic scan. Using the modeless dialog and multithread mechanism, it offers users more controls during the scan process and the same functions like histogram based lookup table, slice to manipulate still image as well as real-time acquired image. Due to the feature of the large size of volume data, the computation of visualization in 3D is often intensive. Finding an efficient rendering algorithm is critical. This thesis implements and optimizes ray cast volume rendering method to visualize volume CT data on PC. The Rendering speed gets improved by using early ray termination, valid projection area check and the boundary box of the dataset on the basic algorithms. Various functions such as histogram lookup table, mapping, region of interest clipping,
normalization, transparency are supported in the tool for users to explore and finely understand the data.
CHAPTER 1. INTRODUCTION

1.1 Computed tomography

Nondestructive evaluation (NDE) technologies are widely used in various industries to inspect the structural integrity and quality of materials and to find potential defects in the components without damaging them. With the evolution of NDE technologies, different methods have been applied such as electromagnetic testing, magnetic particle testing, ultrasonic testing and radiography [1]. Radiography involves the use of penetrating gamma or X-ray radiation to examine materials and parts for imperfections. It is based on the theory that interactions occur as the radiation penetrates matter resulting in a projection of the part geometry into a two dimensional image. The transmitted X-ray flux is measured by a film or a detector. The intensity of the result image shows the internal characteristics of the materials or the parts.

Computed tomography (CT) allows the cross-sectional imaging of an object from a set of X-ray attenuation measurements by illuminating the object from many different directions. The mathematical basis of CT was established by Radon [2] in 1917 who showed that it is possible to determine the value of a function over a region of space if the set of line integrals is known for all ray paths through the region. CT reconstruction requires a large number of computations and was not applied in practice until 1970's, when Hounsfield and Cormack developed the X-ray brain scanner using primitive computers with a reconstruction time of 2 days. They shared the Nobel prize in medicine in 1979 for the contribution to the development of computer assisted tomography (CAT) technology which is now referred to as simply computed tomography. A CT scan system is sophisticated system. In general, it consists of X-ray source, detector, motion control, CT reconstruction and image processing. Due to penetration of X-rays and the sensitivity of absorption cross section to density and atomic number of matter, computed tomography provides a method to generate digital models of the interior and the external surfaces of an object [3]. Multiple successive cross sectional images (slice images) of the object can be stacked up to build a volume of data.
1.2 Data visualization

Whatever inspection methods are used, visualizing data is an important process in NDE systems. Visualization is the process of exploring, transforming, and viewing data as images (or other sensory forms) to gain understanding and insight into the data [4]. Volume rendering is one common method used to visualize volumetric data such as 3D CT scan data.

Through volume rendering, the set of 3D sample data can be transformed into a meaningful 2D image on the computer screen. In this sense, volume rendering shares common functions with computer graphics and image processing. Computer graphics uses surface rendering technology extracting a geometric/topologic representation of an object with geometric surface primitives such as points, lines, triangles, and polygons. The interior part of the object is not described in this process. One of the most common applications of computer graphics is to view the geometry of a part designed on a CAD system. Image processing is the discipline of manipulating an image. It mainly includes image restoration, image enhancement and image compression. Image restoration is the process of taking an image with some known or estimated degradation and restoring it to its original appearance. Image enhancement is used to improve the image appearance to the human eye. Image compression can reduce the massive amount of data needed to represent an image without losing significant information.

The rendering speed of computer graphics techniques is high, but it is very hard to represent the inside world of an object using this method. Unlike computer graphics, volume rendering directly processes the volume data without converting the data to intermediate surface primitive. The rendering process, passing imaginary rays through a three-dimensional object, is very like the process of taking x-ray radiograph. When the rays travel through the three-dimensional data set, they take into account the value of each data element they pass through and keep accumulating these values. When the rays leave the data volume, the accumulated values comprise a two-dimensional data sheet,
which represents the projection of the three-dimensional object. By making certain data elements on the path of ray transparent, the area of interest can be viewed.

Many disciplines and sciences work with three-dimensional data and benefit from volume rendering. These disciplines include nondestructive evaluation, medical imaging, computational fluid dynamics, microscopic analysis, and modeling. Volume rendering can provide ways not only to rotate, zoom, and view these three-dimensional data but also to distinguish one portion from another and focus on the interest part. If sufficient data from a CT scan is available, a model of the inner structure of a part can be built and the potential defects can be analyzed by combining volume rendering and other image processing methods. An example of the application of volume rendering is to study porosity of the soil and obtain a better understanding of how chemicals percolate through the soil based on the CT images of soil samples.

1.3 Motivation and goal of this research

The goal of this thesis is to develop a CT data acquisition program and tools for analyzing and visualizing the CT data sets. The X-ray system is different from other X-ray CT scan systems in our lab. It uses an amorphous-silicon panel as the digital X-ray imager which replaces the X-ray film or image intensifiers and CCD cameras in fluoroscopic applications. The output of the X-ray imager is a 12-bit image with the spatial resolution up to 1536×1920. The motion control indexer is AT6000, which controls four axes (x, y, z, θ) of a Compumotor motor/driver system. The older CT scan system in our lab uses image intensifier. The output phosphor is viewed by a Charged Coupled Device (CCD) camera, which is interfaced to an 8-bit frame grabber. By using image subranging [6], the digitization accuracy of the imager system is 10 ~11-bits. Motion control index uses microprocessor-based indexer PC23. Two versions of XrVision software can control the setup system. The older version of the software consists of two separate programs: the image acquisition program and CT scan program [6] [7]. In the new XrVision software, image acquisition and motion controls are integrated into one program providing a flexible way customizing the scan [8].
The goal of the work in the thesis is to implement general image acquisition, CT scan and automatic scan based on the high resolution X-ray scan system with PaxScan 2520 digital X-ray imaging subsystem, Matrox Meteor-II frame grabber card, Motion control indexer AT6000. The challenge presented in CT visualization is the very large volume of data presented. For example, 100 CT images with the size of 384x384 each would have 14,745,600 sampling points. If the intensity values of the sample points are stored in the form of float data type (i.e. 4 bytes for each sample point), the total file size would be 14,745,600x4 bytes long. The computation needed to visualize 3D data is intensive. This work builds on previous 3D CT data visualization software [9] whose algorithms are not efficient in the volume rendering although some optimization method is used. To render an image of an object with an irregular shape takes an even longer time. Therefore, it is necessary to improve the rendering time to visualize 3D data. Another important issue needing extension is to add lighting and shading to improve reality of the resulting images. This thesis describes a new 3D visualization software, which visualizes 3D data efficiently, improves the reality of the scene by lighting and shading, and integrates new features to manipulate the data such as normalization, ROI clipping, histogram lookup table, slicing the data set.

1.4 Chapter overview

Chapter 2 introduces the basic concepts involved in volume rendering and then discusses two categories of volume rendering methods: image order method and object order method. The implementation of ray cast inverse mapping rendering is discussed in detail. Chapter 3 describes the 3D visualization tool which uses the ray cast method to render the volume data. Chapter 4 details the control software for a high resolution CT scan system including hardware components, software architecture, user interface, and integration test. Chapter 5 summarizes the work and presents conclusions and future work.
CHAPTER 2. VOLUME RENDERING METHODS

2.1 Basic concept: Pixels and Voxels

The data structure of a two-dimensional array can itself be used to represent a two-dimensional image. The value of data elements in the array corresponds to the color or light intensity of the image. These data elements are referred as to pixels. Similarly, a three-dimensional array can represent a volume (see Figure 2.1). Just as the pixels are the elements of 2D image, voxels are the data elements of the data set sampled in three dimensions. The value of the voxels represents the sample intensity.

![Figure 2.1 Voxels and volume](image)

2.2 Perspective and orthographic projections

In computer graphics, projection determines the field of view and how 3D objects are mapped onto a 2D projection plane. Two methods are often used to specify the projection. In perspective projection (see Figure 2.2), the field of view (view volume) is the frustum of a pyramid. The front, back, top, bottom, left and right six clipping planes form the view volume. Objects that fall within the view volume are projected toward the apex of the pyramid where the viewpoint is. The objects that are further away from the viewpoint appear smaller. To get a perspective image on the near plane, we need to cast rays in the shape of a frustum toward the apex. In the orthographic projection (see Figure 2.3), the view volume is rectangular; rays are cast parallel through objects. The distance to the viewpoint does not affect the relative size of the object on the projection plane.
To visualize a 3D dataset, a sequence of processes has to be followed. These steps form the volume rendering pipeline. Figure 2.4 shows the basic steps of volume rendering: filter, gradient computation, resampling, classification, shading and composing. Filtering is the first step, which preprocesses the raw data and extracts the necessary information. Preprocessing may include suppression of noise in the raw data and extracting regions of interest.
The next step after filtering is to find the edges and boundaries between voxels by computing the gradients. Gradients measure how quickly data value is changed in the volume space. It will be used in the shading process. The third step is sampling. First, we cast rays either from the projection plane towards the object or from the object toward the projection plane. Then, we sample the volume along each ray. The sample points determine the values of pixels of the projection image. Interpolation is used to obtain the values of the sample points. Additional details relating to sampling will follow in the next section. After sampling, we assign colors to the sample points depending on their data values after applying a threshold. This procedure is called classification. Shading is used in volume rendering by adding light and illumination to increase the reality and enhance three-dimensional effect of the visualization. The sample points along the same ray will all be projected to the same point on the projection plane. Different composition methods can be applied to these points such as maximum intensity projection, accumulation...
projection, or early ray termination. Using the sampling voxels with their color properties, shading, illumination mode and composition method, the actual rendered color for each pixel on the projection plane is determined.

2.4 Volume rendering method

Volume Rendering is an important part of 3D visualization. The purpose of volume rendering is to provide a way to explore the interior of a volume. No surface mesh is constructed during the rendering process. In general, there are two categories of volume rendering algorithms: object-order method (projection method) and image-order method [4]. In object-order method, voxels are projected onto the projection plane based on their organization in the dataset adding their contribution to the final image. For example, we can traverse the volume from back-to-front or front-to-back, project the voxels onto the image plane and blend the projections into the final image using some composition method. The way image-order method works is similar to digital photograph. We can imagine that there is a view window between the eye and the object that would be rendered. The value of a pixel on the view window is determined by the ray that passes through the pixel and enters into the eye. Image-order rendering is done based on the pixels on the view plane. It casts rays back from the eye and through each pixel on the view window. The voxels encountered along a ray are used to evaluate the pixel that the ray passes through. Before further discussing the volume rendering, we first introduce some fundamental mathematical concepts involved in.

2.4.1 Coordinate and transform matrix

In the computer graphics, a point in the 3D space is represented using a special coordinate system called homogenous coordinates \((x_h, y_h, z_h, w_h)\). The conversion between the homogenous coordinate and Cartesian coordinates \((x, y, z)\) is:

\[
x = \frac{x_h}{w_h} \quad y = \frac{y_h}{w_h} \quad z = \frac{z_h}{w_h}
\] (2.1)
The spatial transform such as translation, scaling and rotation of objects can be represented by an appropriate 4x4 matrix. Recalling homogenous coordinates, \((x_h, y_h, z_h, w_h)\), defined in the previous paragraph, it allows us to perform a series of transforms by repeated matrix multiplication. Perspective and orthographic projections also have their own 4x4 projection matrixes.

The model of the inner structure of the object can be built based on the 3D data set. Often these data sets are the sets of voxels sampled at regular intervals in three major axes. Each voxel has its data value. If we specify the distances that the sample intervals represent and location of one voxel in the volume (often the voxel at the corner), then the original position of the whole volume is given (see Figure 2.5). Then it can be transformed by the operations of rotation, translation, zoom in and zoom out. One 4x4 matrix can be used to transform the volume from its original position to the actual position and projected onto the projection plane.

![Diagram](image.png)

Figure 2.5 Specify the original position of the 3D data set
2.4.2 Forward / inverse projection and transformation.

Assume the original position of a point in the volume has a homogenous coordinate \([x, y, z, 1]\) (definition see 2.4.1). This point is transformed by a series of operations: rotation, translation, and zoom. Each transform can be represented by a 4x4 matrix: \(M_r, M_t, M_z\). After transformation, the homogenous coordinate of the voxel can be calculated in the following equation:

\[
\begin{pmatrix}
  x' \\
  y' \\
  z' \\
  w'
\end{pmatrix} = M_z \cdot M_t \cdot M_r \cdot 
\begin{pmatrix}
  x \\
  y \\
  z \\
  1
\end{pmatrix} = M \cdot 
\begin{pmatrix}
  x \\
  y \\
  z \\
  1
\end{pmatrix}
\]  

(2.2)

\(M\) is called model matrix. After model transform, this point can be viewed using either an orthographic projection or perspective projection. According to [5], the model matrix \(M\) is invertible. This point can be reversed to its original position \([x, y, z]\) using the formulas

\[
\begin{pmatrix}
  x_w \\
  y_w \\
  z_w \\
  w
\end{pmatrix} = M^{-1} \cdot 
\begin{pmatrix}
  x' \\
  y' \\
  z' \\
  1
\end{pmatrix}  
\]  

(2.3)

\[
\begin{pmatrix}
  x \\
  y \\
  z \\
  1
\end{pmatrix} = \begin{pmatrix}
  x_w / w \\
  y_w / w \\
  z_w / w \\
  w / w
\end{pmatrix}
\]  

(2.4)

Let the projection matrix, \(V\), represent the projection. We can combine the model matrix and the projection matrix using a single matrix \(M \cdot V\) called model-view matrix,
to transform the point from its original position \([x, y, z]\) to the projection plane as shown in equation (2.5). \(M \cdot V\) is also a 4x4 matrix, called the transform matrix.

\[
\begin{pmatrix}
x' \\
y' \\
z' \\
w'
\end{pmatrix} = M \cdot V \cdot \begin{pmatrix}
x \\
y \\
z \\
1
\end{pmatrix}
\]

(2.5)

Normalizing the resulting 4x1 vector by \(w\) will get a vector \([x', y', z', 1]\). The coordinate \((x'', y'', z'')\) is the coordinate of the transformed voxel in projection space. For orthographic projection with the projection plane \(Z=0\), the projection matrix is

\[
V = \begin{pmatrix}
1 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1
\end{pmatrix}
\]

(2.6)

From orthographic matrix, \(V\), it can be seen that all the points in the set \([x, y, z]\) where \(z\) can be any value would be projected onto the same pixel \([x, y, 0]\) on the projection plane \(Z = 0\). The projected points still need to be mapped onto the view window (viewport) on the computer screen by either translation or scaling, called viewport transform before it is actually painted. The process transforming an object from its original position to a new position using a model matrix, then projecting it onto the projection plane and, finally, transforming it onto viewport is called forward projection and transform.

### 2.4.3 Splatting

Splatting is a well-known algorithm that belongs to the category of the object order methods [4]. It is like throwing snowball at a window. Each voxel ejects a colored projectile to splat on the projection plane. The projection order is from front to back. The
nearest voxels are splatted on the projection window first. They have the main role in determining the color of the final 2D image since the projection window is between the eyes and volume. To calculate the contribution of each voxel, a 3D kernel weighted by the discrete voxel value is placed around every voxel in the data set and projected onto the image plane. The projected contribution of each voxel is termed the footprint. The final rendering image is obtained by integrating the footprints of all the voxels onto the image plane. When using the splatting method to rendering volume data, a hole might appear in the final image if the radius of the kernel used is smaller than the distance between neighboring voxels.

2.4.4 Ray Cast Inverse Mapping rendering

Splatting uses forward projection through which every voxel in the data set has to be projected onto the projection plane. For a pixel on the projection plane, there are multiple voxels which may be projected onto it. If it is known where these voxels are in the original data set and the data value of these voxels, the corresponding pixel on the projection plane can be rendered according to their data value using some composition method. Ray Cast Inverse Mapping Rendering is one way to solve the problem [5]. Figure 2.6 gives an example to cast ray in inverse mapping through the voxel spaces.

![Ray cast inverse mapping](image)

Figure 2.6 Ray cast inverse mapping
The projection uses orthographic projection. The projection plane is at $Z=0$ and perpendicular to the $Z$-axis. The original position of the 3D data can be set as shown in the left image of Figure 2.6, i.e letting the column and rows of the grids where the voxels are parallel to the three major axes. In this case, it is easy to know which voxels correspond to the pixels on the projection plane. We can render the pixels on the projection plane according to the properties of these voxels. The right image of Figure 2.6 shows the transformed volume using the transform matrix $M$. For an arbitrary orientation it is hard to tell which point in the original data set would be projected to a pixel on the projection plane in this case. For the pixel on the projection plane $(x', y')$, any point in the space with a coordinate of $(x', y', z')$ can be projected onto it, where $z$ can be any value. These points are on the ray which passes through $(x', y', 0)$ and is parallel to $Z$-axis. Since the volume data set is finite, the points in the transformed volume that can be projected on the $(x', y')$ on the projection plane $Z=0$ must have the coordinate of $(x', y', z')$, where $z'$ ranges from some starting value to an ending value. We can sample these points along the ray from a start point and increase $z'$ by a step until the ray leaves the volume as shown in the Figure 2.6. These sample points are called resample points or resample voxels to distinguish them from the sample voxels in the original data set. Now, we know which points in the transformed volume can contribute to the pixel $(x', y')$ on projection plane $Z=0$. To rendering the pixels, we need to know the data value of these sampled points. This problem can be solved by reversing these sampled points from their current positions back to their original position using the inverse of the model transform matrix. After inverse mapping, the resample points have coordinates in the original system. These positions are very likely not on the grid of the original data set. The data values of the resample points can be derived from the surrounding voxels using interpolation, for example, linear interpolation or nearest neighbor. For the resample points on the same ray, different composition schemes can be used along the ray. Figure 2.7 summarizes the inverse mapping procedure. The sample step along $Z$-axis affects the actual rendering speed. If the step is too large, information is lost, while too small a sample step will require more time to implement the inverse mapping than is necessary.
2.4.5 Shading

In the real world, objects appear three-dimensional due to lighting. Lighting effects include light reflection, transparency, surface texture, and shadows. Without lighting, the appearance of objects would be flat. In volume rendering, shading is added to the scene to get better spatial cues and structural information about the data set. An illumination model is used which describes how to assign color to a point in space according to the properties of light, the angle between the viewer and the light, the material properties, and the position of the object. The Phong illumination model is widely used in simulating scenes with lighting. In this model, ambient light, diffuse reflection, and specular reflection need to be evaluated for each point being lit [5].

2.4.5.1 Phong illumination model

Ambient light is also called background light. It comes from reflected lights or from the surfaces of the surrounding objects. It has the feature of coming from all directions at a uniform intensity at all points in the scene. When ambient light strikes a surface, it is reflected in all directions. How much of the ambient light is absorbed and how much is reflected by the object depend on the specific properties of the material. The following
formula describes the color of a point on the surface when it is illuminated by an ambient light with color \( C_a \).

\[ C_o = C_a \times K_a \times O_d \]  

(2.7)

Here, \( C_o \) is the resulting color of the surface. \( K_a \) is ambient reflection coefficient which depends on material properties. It has the value between 0 and 1. For black material \( K_a \) is 0. \( O_d \) is the diffuse color of the point on the surface.

Diffuse Reflection is calculated when an object is illuminated by a point light source radiating in all directions. The light source is also called directional light. Light starts out from the point source. Once it hits the object, it will be reflected in all directions. The calculation of the color at a point in the scene with a directional light needs to take account of another parameter: the direction vector from the light to the point. The direction is shown by \( \vec{L} \) in Figure 2.8.

![Figure 2.8 Diffuse reflection](image)

So in addition to the term that is contributed by the ambient light, we need take into account the diffuse reflection. In this case, the Phong illumination now becomes

\[ C_o = C_a \times K_a \times O_d + C_p \times K_d \times O_d \times \cos(\theta) \]  

(2.8)

Here, \( C_o \) is the color of the point, which is being lit by the light source. \( K_d \) is the diffuse reflection coefficient like \( K_a \). \( \theta \) is the angle between the light source vector and the surface normal at the point. Generalizing equation 2.8 and recalling that the light source normal \( \vec{L} \) and the surface normal \( \vec{N} \), as shown in Figure 2.8, are at angle \( \theta \), we can replace the above formula with:

\[ C_o = C_a \times K_a \times O_d + C_p \times K_d \times O_d \times (\vec{N} \cdot \vec{L}) \]  

(2.9)
The third component in the Phong illumination model is specular reflection (see Figure 2.9). It can create shiny highlights on the surface of an object. Shiny metal has high specular component. When the specular reflection is considered, the full illumination model is:

\[
C_o = C_a K_a O_d + C_p K_d O_d (\overrightarrow{N} \cdot \overrightarrow{L}) + K_s O_s (\overrightarrow{R} \cdot \overrightarrow{V})^n
\]

(2.10)

Here, the \( K_s \) is the specular reflection coefficient and depends on materials, \( \overrightarrow{R} \) is the mirroring normal of \( \overrightarrow{L} \) about the \( \overrightarrow{N} \). \( \overrightarrow{V} \) is the vector from the point viewed toward the viewer. The \( n \) is the specular reflection exponent. \( O_s \) is a separate specular color which is assumed white and doesn't depend on the material’s color at that point.

![Figure 2.9 Specular reflection](image_url)

2.4.5.2 Shading in volume rendering

As noted earlier, in volume rendering, we try to get a better understanding of the data set via visualization. The ultimate goal is not to achieve the realism of the photograph, since the object you are rendering might be invisible in the real world. To reduce the intensity of computation, we will only use one white point light source, with the Phong illumination mode using ambient and diffuse component and no specular component. This mode can provide a good rendering effect [5]. In volume rendering, with no any surface
extracted we have no surface normal available. Hence, the gradient replaces the surface normal of Equation 2.8 to shade volume data.

The gradient is a three-dimensional vector that describes how quickly the intensity changes in a data set. At the edge and boundary between two materials, the gradient would be great. The direction of the gradient gives the orientation of the edge and the boundary. The zero magnitude for a gradient means the voxel is surrounded by the voxels with similar levels of intensity. Therefore it is not on the edge or on the boundary. There are different ways to calculate the gradient. The central difference gradient estimator is a simple and fast method that is commonly used. The main idea is to subtract the intensity of neighboring voxels in each of the direction of the major axes. We use vector $D=[D_x, D_y, D_z]$ to represent the gradient at $(x, y, z)$, then

$$D_x = f(x-1, y, z) - f(x+1, y, z)$$

$$D_y = f(x, y-1, z) - f(x, y+1, z)$$

$$D_z = f(x, y, z-1) - f(x, y, z+1)$$

Here, $f(x, y, z)$ is the intensity of voxel at position $(x, y, z)$ in the data set.

Once the color of each resample point is known, it remains to composite these points along the ray onto one pixel in the projection plane. Each resample point is assigned an alpha value, which represents its transparency. The range of alpha is between 0 to 1. An alpha of value of one means that the point is completely transparent, while the alpha value 0 means it is totally opaque. The opacity value equals to 1 minus alpha. One commonly used composition approach is called front-to-back composition. Assume there are a total of $n$ sample points along the ray, point $a$ and point $b$ are the first and the last sample point on the ray. $I_i$ is the ray intensity at the $i$th sample point. The relationship between $I_i$ and the color of the sample points $C_i$ is:

$$I_i = C_i * (1 - \alpha_i)$$

Here, $\alpha_i$ is the alpha value assigned to the sample point. Let $I(a,b)$ be the ray intensity at the pixel on the projection plane that the ray passes through. Using front-to-back composition, the composition becomes
\[ I(a,b) = \sum_{i=0}^{n} \prod_{j=0}^{i-1} (1-\alpha_j) \]

\[ = I_0 + I_1 (1-\alpha_0) + I_2 (1-\alpha_0)(1-\alpha_1) + \ldots + I_n (1-\alpha_0) (1-\alpha_1) \ldots (1-\alpha_{n-1}) \]

(2.15)

If the accumulated transparency becomes very close to zero at some sample point, any point after it will not contribute anything anymore. Thus, we can stop processing the ray cast. This is called early ray termination.

### 2.5 Comparison between object order rendering and image order rendering

Image order rendering casts a ray back from the viewer’s position through the image and resamples along the ray. It is easier to implement some algorithmic optimizations, for example, early ray termination. However, the resampling introduces random memory access to the dataset and nonuniform mapping of the ray samples to voxels. This contrasts to the regularity of access of voxels in the object order rendering which allows the memory bandwidth to be used more efficiency. It may decrease the memory throughput [10]. Both algorithms and the pipeline can be easily implemented in a parallel way. This is particularly important for processing huge 3D data set (1024\(^3\) or bigger) or real time rendering such as 30 frames per second. Parallel processors and multiprocessor architectures have achieved image generation rates up to 30 Hz on moderate sized data sets [11]. Another alternative way for interactive volume rendering is to use custom architectures with a lower hardware cost and suitable for desktop computer. The VolumePro PCI accelerator board is an example. It has a single chip volume rendering engine configured with 160MB of fast SDRAM which is enough to store 512×512×256 voxel data set. The internal architecture employs Ray cast volume rendering algorithm. It can render a 256\(^3\) volume in real-time [5][12].


CHAPTER 3. 3D VISUALIZATION TOOL

3.1 Introduction

The goal of this work was to find and identify methods for visualization and manipulation of 3D data sets. The previous 3D visualization tool uses OpenGL [13], VC++6.0 and back-to-front object-order rendering method. Voxels are drawn from image plane by image plane. For objects with regular geometry as shown in Figure 3.1, it finds all the 6 surfaces of the objects and only projects voxels on these surfaces onto the projection plane. In this case, the rendering speed is good. However, for irregular objects, it is hard to extract surfaces. Thus every voxel is projected in the program. To process all the voxels of a large data set can cause a considerable reduction in rendering speed. Therefore, a more efficient volume rendering method is needed to solve the problem.

![Figure 3.1 A volume with regular profile](image)

Furthermore, lighting and shading have not been implemented in the old program. Finally, a richer set of features needs to be added to manipulate the 3D data set.

Currently, there are various methods to implement the 3D visualization as discussed in previous chapter. We decided to use ray cast rendering methods for several reasons. The reasons are the following: it is easier to implement the optimization such as early ray termination compared with object order methods. The improvement in rendering speed is especially obvious when the data set is irregular, a situation that is often encountered in
X-ray NDE visualization. To make use of the existing image library, the program was developed using VC++ 6.0 and run on the platform of WinNT and WIN98. Although the computation of volume rendering is intensive, after optimization with early ray termination to the ray cast, the rendering speed is acceptable on PC to process medium size data size like 384x384x100 float CT data. With this tool, users can read two types of data into the memory. The data can be a sequential reconstructed ASCII slice image (the standard output from the CT reconstruction program) or a binary volume data. Various ways are provided to help users to explore the volume data. More details about the tool is in the later part of this chapter.

3.2 Software development

3.2.1 Software architecture

The overall structure of software is based on the MFC multiple document/view structure [14]. Several key classes are used in the 3D visualization software including C3D(rendering), CMdiView(view window), CMdiDoc(fileI/O) and CTAB(users interface), see Figure 3.2.

![Figure 3.2 Software structure for 3D volume rendering](image-url)
All the information (including the rendering parameters and the data set) and methods needed to do volume rendering are encapsulated into the C3D class. Basically, the program supports two types of views: 3D volume rendering view and 2D image view. Each 3D volume render object (an instance of C3D) is associated with a view window (an instance of class CMdiView) where the final projected image is painted.

In the class C3D the rendering parameters are data members, which can be adjusted by the users through various control pages. Examples of these data members include the ROI parameter and mapping range. After transferring these parameters to 3D-volume object, a new volume rendering can be initialized. The CMdiDoc class handles the data import and output between C3D and file system. Another independent class in this program is CTab. It holds the manipulation pages in a property sheet: ROI clipping page, spatial range page, normalization setting page, displaying slice page, intensity range, and view setting page. The controls in these pages directly update the rendering parameter. In addition to the 3D visualization, the classes CimageView, CimgDoc, and CDib provide capability to display 2D image for example BMP, JPG and some basic function on the image including ZoomIn, ZoomOut and pixel trackings.

3.2.2 C3D class

3.2.2.1 Data members

As noted earlier the C3D class encapsulates all the data members and methods needed to do the volume rendering. It has a float pointer, originalData, which points to the beginning memory address of the volume data set. The reason we use float type is to support a wider range of data value in the volume. Structure C_InrTuple is used to store dimension information of the volume. The software provides various ways to control the rendering process. For each manipulation there are associated parameters in the C3D class. For example, variables _ScaleX, ScaleY, _ScaleZ are Scaling factors. The final rendered image depends on the rendering parameters used during the rendering process. There are two import variables that should be mentioned here: HDC ghDC and
BITMAPINFOHEADER bminfoHdr. The pointer bminfoHdr points to a bitmap formatted memory space, which is used to store the final rendering image of the volume. Variable ghDC is the handler of the CMdiView. Using MFC multiple document/view structure, multiple objects can be rendered in the program. So the program must know which window is used to draw a current rendering image of the object. This is done by accessing ghDC.

### 3.2.2.2 Function members

The function members of C3D class can be divided into three categories: IO function, rendering function and user interface function. IO functions handle all the file operations. In the program these IO functions are called by class CMdiDoc when opening, creating or saving a volume. Creating a volume from a group ordered reconstructed ASCII slice files takes a significant amount of time. In order to reduce the space and loading time, a binary VOL file is used to store the whole volume data set. The ASCII slices are stored in the VOL file one slice after another.

The rendering pipeline is implemented in function RenderVolume(). Below it are low-level functions server as normal calculation, matrix calculation, and coordination transform. The rendering algorithm is a ray cast inverse mapping with early ray termination to accelerate the rendering speed and uses orthographic projection. Function Gradient() will calculate the gradients of the sample voxels. In Function Shade(), the gradients and the normal of the light are used to calculate the attenuation to the color of the sample voxels according to Equation 2.8. Once the attenuation coefficients and the original colors (this color is assigned by mapping) of all the sample voxels are known, each pixel on the image plane gets a color with the attenuation coefficient \( \times \) the original color of the sample point projected onto it. One white point light source is applied to the scene. The illumination mode using Phong illumination mode with two components, ambient and diffuse reflection reduces the computational intensity, while achieving a good result in the visualization. Calling displayImage() will raster the whole result 2D image on the appropriate view window.
In the example source code provided in [5], basic rendering algorithm is implemented. After each transform, the spatial range \((z_{\text{min}}, z_{\text{max}}, x_{\text{min}}, x_{\text{max}}, y_{\text{min}}, y_{\text{max}})\) occupied by the boundary box of the volume data is obtained by calculating the coordinates of 8 corners of the boundary box. For each pixel \((x, y)\) on the projection plane, the ray cast starts from \(z_{\text{min}}\) and steps along the \(z\)-axis. It stops either when \(z_{\text{max}}\) is reached or the early ray criteria terminates the process. The basic rendering method provided in [5] uses \(z_{\text{min}}\) and \(z_{\text{max}}\) as a step range for all the rays. Obviously, it is simple but not optimal. This is changed with the result that a significant improvement in rendering speed is obtained in the 3D visualization tool. The way this is done is to recalculate the plane equation for each surface of boundary box. The ray casting only starts at the point where the ray intersects the boundary box effectively reducing the computation time for traversing the volume. For example, in Figure 3.3, for pixel A on projection plane, the ray cast would start from \(z_1\) and end at \(z_2\). An additional method used in the program to improve the rendering speed is to get the polygonal region of the projection of the boundary box on the projection window and only cast ray from the pixels inside the polygonal regions.

The third type of function in the C3D works as a bridge connecting the user input and the rendering pipeline. These functions accept the rendering parameters from a user and accordingly preprocess the volume data or passes the parameters to the rendering
pipeline and initiates a new volume rendering. These functions include ProcessSpatialClip(), Normalization(), TransparentProcess(), Map(), ROIClipProcess(). Map() is a little different from the others in that what it does is just adjusting the lookup table of the projected 2D image.

3.2.2.3 Rendering volume function

This part will discuss how the volume rendering pipeline is implemented in function RenderVolume(). The program always initializes the location of the dataset by letting the lower left corner of the first slice at (0,0,0) in the object space. It uses one white distributed light source with its direction parallel to z-axis. The rendering follows the steps below:

Step 1. Check if the size of the view window is changed. Make sure the object being rendered always appear at the central of the view window. If not, it maps the original point of the projection plane to the central of the view windows.

Step 2. Update rendering parameters: the rotate angles along x-axis and y-axis.

Step 3. Calculate the model transform matrix $M$ for the dataset according to the following transforms: translate the volume from its initial position to the next position with its central at (0,0,0) in object space, scaling transform, rotation transform along x, along y, and translate the volume data making it further away on the z axis to prevent clipping against.

Step 4. Do reverse rotation transform on the light source.

Step 5. Apply the transform matrix $M$ to the boundary box of the whole data set, get its new coordinate of each corner of the boundary box, and the plane equation of each face.

Step 6. Use orthographic projection with the projection plane perpendicular to z-axis. Get the regions where the 6 surfaces of the boundary box will be projected on the projection plane. This is the limit for the areas needed to do the ray cast.
Step 7. Only cast ray from the region on the projection plane where the voxel might be projected. When casting a ray back from (i, j), using the plane equation in Step 5 get the intersections of the ray and the boundary box.

Step 8. Starting from the intersection obtained in Step 7, sample the volume data set along the ray. For each sample point, apply the inverse of the model transform matrix $M$ to get its location before it being transformed. Then interpolate the intensity of the sample voxel using the nearest neighbor. Stop sampling if the ray is outside of the volume data or the sample point is fallen in the region of interesting. Calculate the gradient using the central difference gradient estimator and calculate the attenuation coefficient based on the Phong model. The various rendering parameters from the user should be considered to calculate the gradient and the color. e.g. ROI Clip, transparence.

Step 9. Iterate performing Step 7 and Step 8 for every pixel on the valid area on projection plane.

Step 10. Using lookup table, assign colors for the sample points according to their intensities. Get the final bitmap of the projected image by multiplying the attenuation coefficient with the assigned color for each sample point.

3.3 CT volume data manipulation

3.3.1 Import and output CT volume data

As metioned before, ASCII reconstructed files and binary VOL files can be rendered in the program. Figure 3.4 shows the interface to open the ASCII files. In the dialog window, the user need to specify the file name of the first slice and the total number of slices to be included in the volume. The thumb nail at the left lower corner of the window gives a preview of the slice users select. Similarly, there are two ways to save the volume rendering. If only the rendering image is needed, users can save it as a JPG file. To save the data set and the rendering parameters associated with it, users can go to the
view setting page (see Figure 3.5) and save the data set into a VOL file. The VOL file consists two parts: file header and binary volume data. The file header includes annotation, file name of the first slice, volume size and scaling factor, rotation angle, ROI parameter.

![Figure 3.4 Open ASCII file dialog](image)

![Figure 3.5 View setting page](image)
In the binary part of a VOL file, the scalar value of voxels are stored in float type slice by slice. With the header information, the same volume rendering can be produced directly from the saved VOL file. Although using binary format reduces the space for 3D data, loading large volume data needs a progress bar to show the state of loading (see Figure 3.6). The bar is also used in rendering and calculating the histogram.

![Figure 3.6 Progress bar](image)

### 3.3.2 Data display controls

#### 3.3.2.1 Histogram based lookup table and transparency

A histogram gives a plot of the number of voxels or pixels at each intensity occurring in the data set, see Figure 3.7. The horizontal axis of the histogram represents the range of intensity of the voxels. The vertical axis represents the number of voxels at each value in the data set. From the histogram, we obtain voxel intensity distributions in the dataset and decide how to classify the voxels, namely by assigning an opacity value to a voxel.

![Figure 3.7 Intensity page](image)
We make voxels in a certain range completely transparent in the program. Multiple ranges selection is supported here by using add button and delete button. These multiple selections are stored in a list. Only the voxels with intensity falling into the selected ranges will be rendered. Figures 3.8 are the volume renderings of the same CT data from the soil sample. Figure 3.8 (a) is the resulting image where the soil is rendered. The intensity is between 158-268.38. Porosity is rendered in Figure 3.8 (b) with intensity ranging from 37 to 127.25.38. The number of voxels within the selected intensity range is also shown in the list box.

Figure 3.8 (a) Rendering mass soil
In addition to the IntensityRange Page, the Mapping page (see figure 3.9) also uses the histogram. Color mapping is probably the most commonly used visualization technique in digital graphics. The idea behind it is simple: scalar values are mapped through a lookup table to obtain a color which is applied during rendering to modify the appearance of the point or cell in the final image. Gray scales and rainbow color maps are used in the program. Each lookup table has 256 entries. In the mapping page, users specify the minimum intensity value and maximum intensity value. For the voxels with intensity between the minimum and the maximum value in the mapping page, the value of the $i$th entry will be assigned.

$$i = \frac{(\text{voxel Intensity} - \text{min Intensity})}{(\text{max Intensity} - \text{min Intensity})} \times 256$$  (3.1)
For the voxel intensity beyond the mapping range, the last entry or the first entry will be assigned accordingly. This color multiplies the attenuation coefficient obtained from the Phong model and the actual color applied to the rendering image is determined. The mapping page is based on mapping class - ChistogramDlg [9]. New features are added to the mapping class in this program, namely, the look up table is applied to the histogram allowing the users to see the rainbow color band in the histogram. The vertical scroll bar on the left of the histogram can adjust the vertical axis zoom factor of the histogram. Figure 3.10 also gives an example of two rendered images from the same data set using a rainbow color map but with a different mapping.

Figure 3.9 Mapping page

Figure 3.10 Rendering image with different mapping range
3.3.2.2 Normalization

CT data sets have noise that are contributed by several sources. Some of these are due to drift in the data acquisition process, either from the stability of the X-ray source or from time dependent effects in the detector. These noises can be removed by normalization of various portions of the data that are known to be the same. Figure 3.11(a) is a situation that we have when visualizing 3D datasets.

Figure 3.11: (a) Before normalization (b) After normalization

The data set in Figure 3.11 is a CT scan data with 30 slices 384*384*30. The display shows an interior plane in the data at \( y = 27 \). In reality, the intensity change between slices close to each other should be smooth. We can see a jump at the left image where the red band is. To provide a more accurate data set and a better image, we need to normalize the data to remove this variation between slices. First, we need to select a slice as a reference slice, which in turn is used to normalize all the other slices. The equation is given by:

\[
\alpha_i = \frac{\sum \text{pixel values at } (x, y) \text{ on the reference slice}}{\sum \text{pixel values at } (x, y) \text{ on } i\text{th slice}}
\]

new pixel value at \( (x, y) = \alpha_i \cdot \text{old pixel value at } (x, y) \)
All the pixels in reference slice can be used to calculate $\alpha_i$ during the normalization. This method is simple, but might be easier to lose information especially for the case when there is a rapid change in the data set. The alternate way used here is also simple but requires the users intervention. The user selects which pixels are to be used in the normalization process. In this way, we can make good use of the properties of the data set, for example, the user can select air voxels where it is known that the variation through out the data set is minimal. Here we always assume the slices are stacked up along z-axis. The rendering image in figure 3.11(b) is obtained by the manual normalization. By comparing it with left image we can see the jump area becomes smooth.

In the normalization page (see Figure 3.12), users can use a scrollbar to go through the data set along three major axes. Slices are shown in their original sizes at the left. The Reference Slice button is used to select one slice on the Z axis slice as the reference slice. All the pixels used to calculate $\alpha$ are listed in the list box. In this example, the reference slice is at Z=0. Pixel (91,91) is one of the reference pixels. A small white circle on the left slice can be seen in Figure 3.12 which responds to this reference pixel. Before the entire volume data is modified by the normalization step, users can click the preview button and use the scroll bar to preview the result of normalization one slice after another.
3.3.2.3 Clipping and ROI

Clipping datasets and selecting region of interest are often used in 3D visualization. Two types of clipping functions are supported in this program. One is cropping the data set by specifying the minimum and maximum X, Y, Z coordinates of the boundary box that the ROI occupies as shown in Figure 3.13. A user can specify scale factors of three main axises in the same page. The other clipping function is implemented in the ROI page, see Figure 3.14. In this page, the user can go through slices and select a circular ROI on a slice viewed in the Z direction. This ROI would be applied to every slice in Z direction. Voxels outside of ROI must be padded with a new value or be made transparent in the rendering process. Figure 3.15 and Figure 3.16 show the rendering image under ROI clipping. In Figure 3.15(a), the volume beyond the ROI is completely transparent while in figure 3.15(b) this portion is padded with value zero shown in a blue color.

![Figure 3.13 Spatial range page](image)
Figure 3.14 ROI clip page

Figure 3.15: (a) ROI Clipping without Padding  (b) ROI Clipping with Padding
3.3.2.4 Animation

A very useful tool to aid in visualizing the volume data is animation, where a sequence of images is displayed in rapid succession. These images may vary due to changes in geometry, color, lighting, position, or other rendering parameters. Adjusting these parameters brings a dynamic change in the volume rendering and gives users a new perspective to study and view the data set. This program generates sequential JPG images (frames) which can be used to create animation by some animation tools. Each key frame has rotate parameters and X, Y, Z coordinates of boundary box of the data set. Auxiliary frames is inserted between two key frames. The number of the Auxiliary frames between the key frame can be specified. The more Auxiliary frames are inserted, the smoother change from one key frame to the next key frame. Figure 3.16 shows the animation dialog window. Users can specify the rendering parameters from the control page and rendering the volume. Then the rendering image can be added as a key frame by clicking the add button in the dialog.

![Animation dialog](image)

Figure 3.16 Animation dialog
In the example shown in Figure 3.17, there are two key frames, the only difference between them is: the first key frame has a Ymax of 349, the Ymax of the second one is 10. The number of the auxiliary frames is 4. This configuration will produce 6 animation frames showing the volume is being cut along Y-axis.

![Figure 3.17 A sequence JPG images for animation](image)

### 3.4 Performance evaluation

To evaluate the performance of the 3D-visualization tool, we test the software on a Dell computer with 128M RAM and a CPU speed of 500MHz. The result shows the rendering time varies with the size of the data set, the orientation and the ROI of the sample. For a soil sample of dimensions 384*384*50, the rendering time varies from 1 second to 4 seconds when ROI is the whole volume. For a sample with dimension 384*384*99, the rendering time changes between 2 and 8 seconds.

When we work with the tooth sample at higher resolution 512*512*80, the rendering speed decreases greatly. The rendering time can be as high as 39 seconds. During the rendering process, the program is frequently reading and writing data from and to the disk. The exchange data between the memory and the disk greatly increases rendering time. This happens due to the fact that the memory requirement for running rendering
tooth sample exceeds 128M that the computer is capable of providing. To see this, recall that the Windows operating system uses the virtual memory mechanism, which separates the physical memory from the user logical memory [14]. The virtual address space that the operating system can provide is only related to the processor address size. The Windows operating system running on a processor that has 32-bit address bus will have the total virtual space of $2^{32} = 4\text{Gbytes}$. The basic unit of the virtual memory is a page with size of 4K. The operating system allocates pages in virtual memory to hold the program, data and the resources for processes. When the program actually attempts to execute or read/write from a page not in the memory, the attempted memory access triggers a page fault exception. The operating system then moves some pages back to disk and allocates the physical memory for that page and brings the content from disk to that physical memory. In the visualization program each voxel has a float type variable to hold the intensity and a byte type variable to hold its rendering propriety. The memory demand for the whole tooth sample is at least 104M. Therefore, we see the frequent data swap between the disk and program during the rendering process.

Larger datasets require a longer time to render. Making some portion within certain intensity range transparent can cause the irregularity of the datasets which increases the rendering time. Because we use early ray termination and valid inverse mapping area check, the rendering speed is still much better that the old program. A comparison of the rendering speed in the case of irregular volume data with the same CT data shows that the old program takes 22 seconds to render $20\times384\times384$ CT data while the new tool can finish rendering in 2.5 seconds. The test on 30 slices shows a similar result. When rendering a regular volume data, the rendering speeds in both cases are fast.
CHAPTER 4. HIGH RESOLUTION CT DATA
ACQUISITION SOFTWARE

4.1 Introduction

In the previous chapter the visualization of volume data is discussed. This chapter deals with the acquisition of CT data. The software design for a high resolution X-ray system is described. Figure 4.1 depicts the system.

In the system, an X-ray fan beam is projected through a sample at different angles by rotating the sample in a small angular increment (see Figure 4.2). The intensity of attenuated X-ray is measured by the detector. The raw data set made up from a projection for all orientations is referred to as a sinogram. Figure 4.3 shows an example of the structure of a sinogram, which contains the projections of one cross section of the sample. The cross sectional image (slice) is reconstructed from the sinogram using fan beam projection algorithm [9].
4.2 System description

4.2.1 Digital X-ray imaging subsystem PaxScan 2520

The hardware components of the system include the X-ray source and its control unit, the sample positioner and its control unit, the detector consisting of the PaxScan 2520 digital X-ray imaging subsystem [15] and digital Matrox Meteor-II frame grabber card and control PC, see Figure 4.1. The digital imaging subsystem distinguishes this
system from the other CT scanners commonly available. The amorphous Si panel offers better dynamic range, automatic correction, higher image resolution up to 1536×1920 pixels. It also offers a significant improvement in term of image quality over other CT detectors.

The detector subsystem has three main components: the receptor, which houses the solid-state, flat panel sensor, the command processor, and the power supply. The core of the detector is an array of amorphous silicon p-I-n photodiodes and thin film transistors (see figure 4.4).

![Figure 4.4 Internal configuration of the receptor](image)

On top of the (a-Si) array is an X-ray scintillator, which converts the X-ray photons to visible radiation. The visible photons are absorbed by pin photodiodes and converted to electron hole pairs. The pin photodiodes are discharged when the pixel’s TFT is turned ON. The charge is collected by an integrating amplifier and converted to a voltage, which is digitized by 12 bit analog-to-digital converters. The amorphous silicon has a number of features that make them well-suited to X-ray sensor applications. It can be deposited over a large area onto glass substrates and has a very low pixel-to-pixel crosstalk. With an appropriate selection of X-ray scintillator it can image a wide range of X-ray energies in diagnostic imaging and non-destructive testing application. Compared with a-Si detector, a CCD-based imaging system offers the substantial loss in signal quality due to the relatively small solid angle in which light is collected. The larger the area to be imaged, the greater the loss.
The command processor for the image subsystem consists of an embedded computer, image correction hardware, and external hardware interface connections. It can be controlled via the Ethernet interface or two serial ports. The command processor allows real time correction of images and 30 frames per second 16-bit digital video output. The reason for calibration is the non-uniformity of the receptor and the differences between the readout amplifiers resulting in the background and gain variations. Correction is also needed for defective pixels. The image acquisition modes currently supported are shown is the Table 4-1.

<table>
<thead>
<tr>
<th>Mode</th>
<th>Frame rate (Hz)</th>
<th>Frame Size</th>
<th>Acquisition Type</th>
<th>Pixel Binning</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fluoroscopy-Normal</td>
<td>30</td>
<td>768x960</td>
<td>Continuous</td>
<td>2x2</td>
</tr>
<tr>
<td>Fluoroscopy-Full Resolution</td>
<td>7.5</td>
<td>1536x1920</td>
<td>Continuous</td>
<td>1x1</td>
</tr>
<tr>
<td>Radiography</td>
<td>7.5</td>
<td>1536x1920</td>
<td>Accumulation</td>
<td>1x1</td>
</tr>
</tbody>
</table>

4.2.2 Software layer of digital X-ray imaging subsystem

The software architecture controlling the digital X-ray imaging subsystem PaxScan 2520 consists of four levels as shown in Figure 4.5. The manufacturer provides the Ethernet Interface function for users to communicate with the command processor via a Win32 dll. The class, CvivaDetector, based on the lower level interface function provides a programming interface for controlling the detector. The common operations we might use are encapsulated into the class. Users can simply initialize their command by one function call and do not need to know detailed procedures involved inside the function.

Basic operations to the PaxScan 2520 in CvivaDetector is following:

Connection control Functions: int OpenLink() and int CloseLink(). This pair function allows the user to establish or terminate the Ethernet connection between the host.
computer and the command processor. The returned value indicates success or failure of the function call. OpenLink() is used before initializing an operation command and CloseLink() is always called when an operation command is over.

**Mode Control functions:** the system can be configured to different modes. The manufacturer provides a separated configuration utility named Config.exe to build and load the necessary configuration files (system configuration file and receptor configuration file) for the image system. In our CT data acquisition program, we use Fluoroscopy-Normal and Fluoroscopy-Full Resolution as recommended by the manufacturer. Void GetModelInfo() gets the information in all the modes that the system can be supported. This information includes the number of modes available in the command processor and the description of each mode. At most three modes can be loaded into the system and only one mode can be active at a time in the system. CString GetCurrentModeStr() get the description of the current active mode. int SetMode(int i) sets the current active mode among the modes loaded in the systems.
Calibration control functions: int DisableAutoOffset() and int OffsetCalibration(). In the default calibration for the system, offset calibration will occur every five minutes. Since offset calibration requires that the X-ray beam be off, we provide two functions DisableAutoOffset() and OffsetCalibration() for controlling when an offset calibration to be done.

The X-ray images can be outputted either from the Ethernet port of the command processor as still images or from its 16 bit digital video port as real time images. Accordingly, two types of image acquisition functions are provided in the program: CvivaDetector::GetImage(bool integration, int frameNum) acquiring images directly from the detector and CFrameGrabber::AccumGrab(bool integration, int frameNum) acquiring images from the frame grabber.

Image acquire functions directly from the detector: Different image resolution is supported in different modes. Function CSize GetImageSize() returns the size of image current active mode can support. This function must be called every time the active mode is changed. Since image size corresponds to the memory requirement, confusion of the image size is fatal to the program. Function unsigned short *GetImage(bool integration, int frameNum) provides a way to acquire still images from the Ethernet port. Parameter frameNum specifies the number of images to be acquired. Parameter integration in the function determines the operation of the acquired image: integrating or averaging. This function returns an unsigned short pointer which points to the memory allocated to store the final image.

Acquire control functions: Utilizing the 16 bit digital video output of PaxScan 2520 and the frame grabber, real time X-ray image can be captured. So CvivaDetector also provides int Acquire() and int EndAcquire() which work with the frame grabber function to capture real time image. Acquire() simulates the hardware handshaking signal, informing the command process X-ray machine is ready. Image acquisition starts after the valid x-ray signal is received. During image acquisition, a real time image is sent out from the 16-bit digit video port of the PaxScan 2520. EndAcquire() sends a handshaking signal to terminate the acquisition.
Reset() function: allows user to abort any incomplete acquisition or calibration and the command processor will return to the default mode.

4.2.3 Software layer for image acquisition from frame grabber

Class CFrameGrabber provides another interface using digital Matrox Meteor-II frame grabber card to acquire real time image [16]. Figure 4.6 shows the relationship with the CFrameGrabber in the program. The Matrox Imaging Library, the interface to control the grabber card, is provided by the manufacturer.

![Figure 4.6 Software layers of image acquisition from frame grabber](image)

Compared with function GetAccImage in CvivaDetector, CFrameGrabber directly captures the digital video output from the command processors. It has the capability to capture real time images, however, only fluoroscopic acquisition can output the digital video. Other images such as the defect map must be retrieved from the detector. The following gives the basic functions in the CFrameGrabber.

**Function Initialize():** Allocate all the memory needed for capture the 16 bit digital video and initialize the card to proper state.

```c
unsigned short * AccumGrab(bool integration, int frameNum); acquires a certain number of x-ray images. The input parameters and returned value are same as the GetAccImage function in CvivaDetector.
```
void NotStopAcq(unsigned short *addr, int x, int y): capture images from the video input port of the frame grabber card. The parameter x and y specify the size of image to be captured. The captured image is stored at the buffer pointed by the parameter pointer addr. This function works with Acquire() and EndAcquire() in CvivaDetector to implement the real time image acquisition.

4.2.4 Motion control indexer (AT6000)

AT6000 is a microprocessor-based indexer, which can control four axes of any size Compumotor motor/driver system [17]. The indexer occupies one ISA AT slot. Figure 4.7 shows the software layer of AT6000. Class CAT6000 developed by Huafu Lu encapsulates the common operations on the AT6000 indexer.

![Figure 4.7 Software layers of AT6000 indexer](image)

Windows NT and Windows 2000 do not allow an application to directly access an IO port, so a device driver is used to communicate with the code on the indexer. ASCII commands to control the motion are sent through the software layers and interpreted and executed by the on-board program. For detailed descriptions on how to use the CAT6000, see [18].
4.2.5 The overall structure of the software

The features of the high resolution CT (HRCT) data acquisition program include integration of the amorphous silicon detector, motion control and a user interface in a WinNT operation system. The software was developed with VC++6.0 using MFC document/view structure. CImage provides almost 50 functions to supporting reading, saving and displaying images in various formats [8]. In the HRCT programs, the images can be loaded from the disk, acquired from the frame grabber or Ethernet port. Images can be either still images or real time images. In despite of these differences, all images are treated in the same way in respect of manipulation: saving, zoom in, zoom out, pixel tracking, slice, and histogram lookup table. This benefits from the objected orient design and modeless dialog we used during the design. In this way, we combine three class Cimage, CHRCTView, CHRCTDoc to handle various image objects and their manipulations. Compared with the overall structure of the new XrVision which distinguishes images according to their sources, the HRCT structure makes the program more efficient, easier maintained while offering the same manipulation functions to all sorts of images.

The class, ChardWare, handles the device control in the X-ray system. The classes, CVivaDetector detector, CFrameGrabber frameGrabber and CAT600* m_pMotor are the instances of hardware devices. Besides image acquisition, the program supports both CT scan and automatic scan. The functions such as pausing and resuming are provided to control the scan progress. HRCT implements these functions using the data member CWinThread *scanThread in the CHardWare.

4.3 Implementation of the software

4.3.1 Acquisition X-ray image

PaxScan 2520 can load at most three modes into the command processor. Only one mode can be active at any time. Figure 4.8 shows the image acquisition control page. It
lists the modes that are loaded in the image systems. The detector mode supports different image resolution and frame rate. Users can specify the desired mode and image source. X-ray images can be acquired in a continuous way up to ten frames per second (real time image) or acquired until reaching a certain number of frames specified. For the latter, acquired images can be either averaged or integrated into one image to reduce the random noise produced by the detector or frame grabber. The use of threads allows the program to accept and respond to user input while acquiring image. The progress of the acquisition is shown by the progress bar.

4.3.2 Xray image manipulation

The operations on images in New XrVision program such as histogram and slice are supported in the HRCT program, see Figure 4.9. What's new is that it applies modeless dialog to implement the slice function so that users can use the multiple tools to analyze the image at the same time. The previous program uses modal dialog boxes which
require the user to respond the dialog before continuing the program. Modeless dialog boxes stay on the screen and are available for use at any time and permit other user activities. To create a modeless slice, the public constructor SliceDlg::CSliceDlg(CWnd* pParent) should be called and then Create member function is called to load the dialog resource. To destroy the modeless dialog, DestroyWindow is called to destroy the dialog-box window. This is followed by calling function PostNcDestroy to release the memory.

![Figure 4.9: (a) X-ray image (b) Histogram (c) Slice](image-url)
Zoom in and Zoom out (Scaling) are the basic function of the image programs. In previous image programs, CScrollView is used to display the image. A simple call to the Window API function StretchBlt() can implement the zoom function. The disadvantage of the old method is the upper left corner of the view window always displays the left corner of the image. So after several zoom in operations, it is possible that the region in which a user wants to focus on is clipped out by the view window. At this time, users need use the scroll bar to move the ROI into the view windows. The example is shown in the Figure 4.10. Considering that the actual acquired image size may be up to 1536×1952, while a common monitor can display 1280×1024 pixels, the HRCT fixes the window size and make the user's point of interest to be the central part of the view windows after each zoom operation. Thus the region of interest can be displayed on the view window.

![Figure 4.10 Two sorts of implement methods for scaling functions](image)

4.3.3 Motion control

The sample position can be adjusted from 4 directions by translating along X, Y, Z-axis and rotating around Z-axis, see Figure 4.11. In this motion control system, optic travel limit sensors are used to detect when the sample reaches the limiting point or is too close to the X-ray source. The home sensor provides a fixed point on the linear guide of
positioner to which the carriage can be commanded to return repeatedly[19]. This fixed point can be used a reference point to set up an absolute coordinate.

The software motion control page allows users to control the movement of the sample positioner along an individual axis. The interface shown, in Figure 4.12, is very similar to the motion control page in the new XrVision program. Additional information about this page can be found in [8].
4.3.4 CT scan

In the HRCT program, the CT scan setup page is used to set the scan based on the setup image, see Figure 4.13(a). The parameters for scan are shown on the CT Scan Setting page, see Figure 4.113(b). The CT Data can be collected in two ways either in bands or in full image. A band consists of a group of slices of successive cross sections of the sample. In the setup image, the blue line is used to mark the location of the beginning cross section of a band. The white lines following the blue represent the remaining cross sections in the band. The slices of cross sections in a same band are collected together at the same time as the sample rotates through the CT scan.
When the sample is too long in the Z direction to fit in the field of view, multiple bands maybe used to set up the scan. Thus, after acquiring all the slices for a band, the sample moves along the Z axis to make the cross sections of next band entering the detectable area of the system and starts collecting the new band.

The remaining parameters of the page (see Figure 4.13(b)) are used to specify acquire mode for the scan and how to save the data. After clicking the start scan button, the scan begins and the active page will switch to Scan control page, shown in Figure 4.14. The CT control page displays the progress of the scan that is on going. The state of each step in the scan will be updated in the status window. The multiple thread mechanism gives the program the capability to control the progress of the scan such as pause, cancel, and restart.

Figure 4.14 CT control page
4.3.5 Automatic scan

The HR_CT program integrates the automatic scan function. The scan might be 1D, 2D, or 3D depending on how the user specifies the command items in the automated scan page. Figure 4.13 gives an example of defining 1D scan along z-axis. Automatic scan and CT scan have common in that they both need to show the status of the scan and a user might need to interrupt the scan progress. So HR_CT program uses the same scan control page to meet these requirements. Each scan has its own controlling function for the scan thread. The controlling function is responsible to do the scan and exchange information such as scan status, interface activity from the scan control page. This kind of structure makes codes more concise and more efficient.

![Automated Scan Setting](image)

Figure 4.15 Automated Scan Setting
4.3.6 System Calibration

The system calibration page (see Figure 4.16(a)) includes parameters used during the scan process and the CT reconstruction. These parameters are related to a particular CT configuration. Hence, it is necessary to do calibration before starting a new scan.

The pixel on which the axis of rotation projects is called the center of rotation. Center of rotation (COR) (see Figure 4.17) is needed by the CT reconstruction algorithms and is very important for high resolution CT image. The incorrect measurement of COR location will cause image degradation. Ideally, center-of-rotation should be collinear with the midline of the fan beam. In practice, a constant misalignment can occur when the X-ray source, the sample or the detector array is mispositioned. The calculation of actual COR can be done using the projections of an objection at 0° and 180°. The edit window for COR shows the location of the detector pixel of COR..

![Figure 4.16: (a) system calibration page (b) Image of dead pixels](image-url)
Pixel size tells the users the practical size that the pixel on the X-ray image represents. Offset position is the default position of the positioner related to the home switch. When users click the home button on the motion control page, the sample will move to the default position. Alignment row is the projection of the center of the X-ray fan beam. It helps users to define the location of a band during the setup.

The amorphous silicon panel has a finite number of dead pixels and lines, using the image of dead pixels button can retrieve the image from the command processor. Figure 4.16(b) is an example of the image highlighting dead pixels. In the image, the white band on the left edge is reserved by the manufacturer, and the segments of the white line consist of dead pixels. The command processor removes the dead pixels by the interpolation between the pixel values of the nearest neighbors.
4.4 System demonstration

A CT scan on a part made of alumni examining the porosity inside it will illustrate the use of the system. The collected sinogram will be reconstructed and visualized by 3D visualization tool. The specimen is a tensile test sample with its diameter increasing towards the two ends. During testing, the coupon breaks at the center, so any metal defects such as pores need to be identified. The central part of the specimen is examined. The setup for the CT scan is shown is Figure 4.18.

![Scan setting for test](image)

Figure 4.18 Scan setting for test

Figure 4.19 shows the specific setting for the scan parameters. According to the setting, the slices of 150 cross-sections are acquired every 2 degree from the frame grabber. The size of the X-ray image is 768×960. The voltage of the X-ray source is 49.7keV and the current is 0.319mA. The interval between 2 cross-sections is 10 pixels,
A total of 150*180 projections are collected and saved into 15 sinograms each consisting of ten slices. Figure 4.20 shows one of the sinograms. The sinuous curves in the sinogram shows some of the features of the specimen. It is hard to directly relate the sinograms to the cross sections being scanned. After running 2D CT reconstruction program on the sinograms, the ASCII files containing the 150-slice images are reconstructed as 256x256 images. Figure 4.21 shows the slices images at the center of the coupon. In the slice image, the area inside the white circle represents cross section of the sample and the dark region in it is where the bubble is.
Figure 4.20 Sinogram of 10 cross sections

Figure 4.21 Reconstructed cross sectional images from sinogram
The stack of the 150 CT slices is shown in figure 4.22. Aluminum and air have different X-ray absorption coefficients. The difference shown in the volume data is that the scalar value of aluminum voxel is higher than that of the air and bubble. For this test, the intensity range of the air and bubble is from -83.58 to 129.93 while the aluminum has a range of 129.95 to 358.09. This can be viewed clearly using the zoom function of the left scrollbar on the histogram. According to this, Figure 4.23 is the rendered image showing only aluminum. Holes can be seen in the surface of the specimen. The rendering of the bubbles in the sample is shown in figure 4.24. Color Mapping, ROI clipping, and transparence make it easy to extract the bubbles from the sample.
Figure 4.23: (a) Rendering the entire coupon  (b) Cut the coupon at $y=120$
Figure 4.24 Visualizing the porosity of the part
CHAPTER 5. CONCLUSIONS AND FUTURE WORK

5.1 Conclusions

In this thesis, a 3D-visualization tool and control software for a high resolution X-ray data acquisition system were developed. The 3D-visualization tool provides various approaches to analyze and visualize the volume data. Due to the large size of the dataset, the computation associated with visualization in 3D is often intensive. Finding an efficient rendering algorithm is critical. This thesis implements and optimizes ray cast volume rendering to visualization volume data on PC. It uses early ray termination, valid projection area check and the boundary box of the dataset to optimize the basic volume rendering algorithms. The rendering speed for 3D datasets with an irregular shape is improved greatly over the old program. Various functions such as histogram lookup table, mapping, region of interest, clipping, normalization, transparency are supported for users to explore and understand the data. In practice, it is now used to analyze the porosity of the soil samples.

The high resolution X-ray scan system of the X-ray group at Center of Non-destructive Evaluation consists of motion control, X-ray source, detector, and frame grabber. The new amorphous silicon detector is used to replace the image intensifier and CCD camera and provides the high-resolution X-ray image. In this thesis, the corresponding control class for this detector is developed in VC++. It encapsulates the necessary functions to control the detector and provides a convenient interface for application development. The development of control software, HRCT, for the scan system integrates the features of the two versions of XrVersion whose hardware system has different motion control index and detector. In HRCT, image acquisition, CT scan, automatic scan are integrated into one program. Using the modeless dialog and multithread mechanism, HRCT offers users more control during the scan process and functions such as histogram based lookup table and the ability to manipulate still images as well as real time acquired images.
5.2 Future work

Both 3D visualization tool and high resolution X-ray scan software need enhancement in some areas as described in the following:

- **3D visualization tool**
  It can be optimized furthermore to improve the rendering speed using methods such as the octree method. It would be very efficient when the actual space the rendering object occupies is much smaller than the boundary box. For huge data sets, for example $1024 \times 1024 \times 1024$, implementing rendering computation on a parallel computer will have to be considered. After parallel computation, the resulting image data can be sent to a front-end program, which runs on a PC while has a user interface and displays the rendered image. The data can be exchanged through the network using the TCP/IP protocol.

- **High resolution X-ray scan software**
  Offset calibration must be done with the X-ray unit off. For the HRCT program, using RS232 interface to control X-ray source can be included to implement the automatic offset calibration during the long time scan. The old format is ASCII and only allows 10 slices in one sinogram. Redefining the sinogram format of the output of HRCT is needed to save them more efficiently on the disk.
BIBLIOGRAPHY

[9] Wei Yan, 3D CT Visualization, creative component, Iowa State University, Ames, IA, 2000