Analysis of Very Large Scale Image Data using Out-of-core Technique and Automated 3D Reconstruction using Calibrated Images

Chandrashekara Hassan Raju
Wright State University

Follow this and additional works at: https://corescholar.libraries.wright.edu/etd_all

Part of the Computer Sciences Commons

Repository Citation
https://corescholar.libraries.wright.edu/etd_all/177

This Thesis is brought to you for free and open access by the Theses and Dissertations at CORE Scholar. It has been accepted for inclusion in Browse all Theses and Dissertations by an authorized administrator of CORE Scholar. For more information, please contact library-corescholar@wright.edu.
ANALYSIS OF VERY LARGE SCALE IMAGE DATA USING
OUT-OF-CORE TECHNIQUE AND AUTOMATED 3D
RECONSTRUCTION USING CALIBRATED IMAGES

A thesis submitted in partial fulfillment
Of the requirements for the degree of
Master of Science

By

CHANDRASHEKARA HASSAN RAJU
B.E., Visvesvaraya Technological University, India, 2004

2007
Wright State University
I HEREBY RECOMMEND THAT THE THESIS PREPARED UNDER MY SUPERVISION BY Chandrashekara Hassan Raju ENTITLED Analysis of very large scale image data using out-of-core technique and automated 3D reconstruction using calibrated images BE ACCEPTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF Master of Science.

Thomas Wischgoll, Ph.D.
Thesis Director

Thomas Sudkamp, Ph.D.
Department Chair

Committee on Final Examination

Thomas Wischgoll, Ph.D.

Arthur Goshtasby, Ph.D.

Thomas Hartrum, Ph.D.

Joseph F. Thomas, Jr., Ph.D.
Dean, School of Graduate Studies

ABSTRACT

Analysis of large scale volumetric data sets is very important from the researcher's and technological point of view as it helps in understanding and analyzing the original data. For example, when analyzing volumetric images from cast arteries of pigs, it is essential to apply a suitable segmentation in order to retrieve morphometric data from the scanned image. Due to improved scanning technology, such as MicroCT, these scanned images can get as large as several gigabytes in size. In order to analyze such large scale data sets on common desktop computers is a challenge due to limited main memory. By using out-of-core techniques, where the hard disk is used as main storage medium while the main memory serves as cache, it is possible to process any size data efficiently on a off-the-shelf PC with limited memory. Filters have been implemented and are applied to the original data to process and transform the original data into another form which can be further analyzed. The presented method is not limited in terms of data set size. The data set size that can be processed is only limited by the size of the hard disk. Hence, the novelty of the described technique is the ability to apply the implemented filters to data sets of almost unlimited size using common, off-the-shelf desktop computers.
In the second part of the thesis work, an effort has been made to automate the 3-D reconstruction of bi-planar images using epipolar approach. The standard approach would be to find the matching features in both the images called corresponding points. Then, from such a correspondence, depth can be easily calculated using standard triangulation method. This type of classical method would require careful selection of the matching features. Here we propose a technique which does not involve selection of matching features which requires manual intervention thereby automating the process. This approach produces reasonable results for the calibrated images due to the projection.
# TABLE OF CONTENTS

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. INTRODUCTION</td>
<td>1</td>
</tr>
<tr>
<td>Motivation</td>
<td>1</td>
</tr>
<tr>
<td>1.1. Analysis of Large Scale Volumetric Data</td>
<td>1</td>
</tr>
<tr>
<td>1.1.2. 3-D Reconstruction using Calibrated Images</td>
<td>2</td>
</tr>
<tr>
<td>1.2 Thesis Objective</td>
<td>3</td>
</tr>
<tr>
<td>1.3 Thesis Organization</td>
<td>4</td>
</tr>
<tr>
<td>2. GENERAL CONCEPTS</td>
<td>5</td>
</tr>
<tr>
<td>2.1 Image Processing</td>
<td>5</td>
</tr>
<tr>
<td>2.2 General Definition of a 3D Dataset</td>
<td>6</td>
</tr>
<tr>
<td>2.3 A Brief Overview of Computer Vision</td>
<td>8</td>
</tr>
<tr>
<td>2.4 Some Imaging Technologies</td>
<td>9</td>
</tr>
<tr>
<td>2.4.1 X Rays</td>
<td>9</td>
</tr>
<tr>
<td>2.4.2 CT (Computed Tomography)</td>
<td>10</td>
</tr>
<tr>
<td>2.4.3 MRI (Magnetic Resonance Imaging)</td>
<td>11</td>
</tr>
<tr>
<td>3. RELATED WORK</td>
<td>12</td>
</tr>
<tr>
<td>4 ANALYSIS OF LARGE SCALE IMAGE DATA USING OUT-OF-CORE TECHNIQUE</td>
<td>17</td>
</tr>
<tr>
<td>Section</td>
<td>Page</td>
</tr>
<tr>
<td>----------------------------------------------</td>
<td>------</td>
</tr>
<tr>
<td>4.1 Image Processing</td>
<td>17</td>
</tr>
<tr>
<td>4.2 Analysis of Data</td>
<td>19</td>
</tr>
<tr>
<td>4.3 Out-of-Core Technique</td>
<td>20</td>
</tr>
<tr>
<td>4.4 Memory Mapping</td>
<td>21</td>
</tr>
<tr>
<td>4.4.1 Memory Mapping Function</td>
<td>21</td>
</tr>
<tr>
<td>4.5 Advantages of Memory Mapping</td>
<td>25</td>
</tr>
<tr>
<td>4.6 Limitations of Memory Mapping</td>
<td>27</td>
</tr>
<tr>
<td>4.7 Implementation and Experimental Results</td>
<td>27</td>
</tr>
<tr>
<td>5. 3-D RECONSTRUCTION USING CALIBRATED IMAGES</td>
<td>37</td>
</tr>
<tr>
<td>5.1. Description of Epipolar Concept</td>
<td>37</td>
</tr>
<tr>
<td>5.2. Intersection of 2 lines in 3D</td>
<td>42</td>
</tr>
<tr>
<td>5.2.1. Algorithm</td>
<td>42</td>
</tr>
<tr>
<td>5.3. Setup and Results</td>
<td>46</td>
</tr>
<tr>
<td>5.3.1 Centerline Algorithm</td>
<td>50</td>
</tr>
<tr>
<td>5.3.2 Normal Vector</td>
<td>52</td>
</tr>
<tr>
<td>5.3.3 Results and Snapshots</td>
<td>55</td>
</tr>
<tr>
<td>6. CONCLUSION AND FUTURE WORKS</td>
<td>65</td>
</tr>
<tr>
<td>6.1 Conclusion</td>
<td>65</td>
</tr>
<tr>
<td>6.2 Future Works</td>
<td>65</td>
</tr>
<tr>
<td>7. REFERENCES</td>
<td>66</td>
</tr>
</tbody>
</table>
**LIST OF FIGURES**

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1</td>
<td>Relation between Computer Vision, Image Processing Various Other Fields</td>
<td>6</td>
</tr>
<tr>
<td>4.1</td>
<td>Pig Heart with Air Bubbles</td>
<td>29</td>
</tr>
<tr>
<td>4.2</td>
<td>Pig Heart with Air Bubbles Removed</td>
<td>29</td>
</tr>
<tr>
<td>4.3</td>
<td>Graph showing the time performance analysis for Gaussian Filter</td>
<td>32</td>
</tr>
<tr>
<td>4.4</td>
<td>Graph showing the time performance analysis for Threshold Filter</td>
<td>34</td>
</tr>
<tr>
<td>4.5</td>
<td>Time comparison chart between ITK’s Gaussian filter and our method</td>
<td>35</td>
</tr>
<tr>
<td>4.6</td>
<td>Time comparison chart between ITK’s threshold filter and our method</td>
<td>35</td>
</tr>
<tr>
<td>5.1</td>
<td>Concept of Triangulation in Stereo Images</td>
<td>39</td>
</tr>
<tr>
<td>5.2</td>
<td>Epipolar Geometry</td>
<td>41</td>
</tr>
<tr>
<td>5.3</td>
<td>Showing two points $P_a$ and $P_b$ on lines $P1P2$ ad $P3P4$</td>
<td>43</td>
</tr>
<tr>
<td>5.4</td>
<td>Environment set up showing the virtual cameras and image plane</td>
<td>48</td>
</tr>
<tr>
<td>5.5</td>
<td>Pictorial representations showing a bunch of 3D points</td>
<td>50</td>
</tr>
<tr>
<td>5.6</td>
<td>Centerline for a branch of phantom wire</td>
<td>51</td>
</tr>
<tr>
<td>5.7</td>
<td>Direction Vector for a Point on the Boundary</td>
<td>53</td>
</tr>
<tr>
<td>5.8</td>
<td>Left View of Phantom wire</td>
<td>55</td>
</tr>
<tr>
<td>Section</td>
<td>Description</td>
<td>Page</td>
</tr>
<tr>
<td>---------</td>
<td>-------------</td>
<td>------</td>
</tr>
<tr>
<td>5.9</td>
<td>Right View of Phantom wire</td>
<td>56</td>
</tr>
<tr>
<td>5.10</td>
<td>Left View of Phantom wire after Rendering 3D Points</td>
<td>57</td>
</tr>
<tr>
<td>5.11</td>
<td>Right View of Phantom Wire after Rendering 3D Points</td>
<td>57</td>
</tr>
<tr>
<td>5.12</td>
<td>Shows the Extracted 3D View of Left Image</td>
<td>58</td>
</tr>
<tr>
<td>5.13</td>
<td>Shows the Extracted 3D View of Right Image</td>
<td>58</td>
</tr>
<tr>
<td>5.14</td>
<td>Graph showing how the Diameter varies along the Length</td>
<td>59</td>
</tr>
<tr>
<td>5.15</td>
<td>Left View of Real Medical Data</td>
<td>60</td>
</tr>
<tr>
<td>5.16</td>
<td>Right View of Real Medical Data</td>
<td>61</td>
</tr>
<tr>
<td>5.17</td>
<td>Left View of Real Image after Rendered using 3D Cloud Points</td>
<td>62</td>
</tr>
<tr>
<td>5.18</td>
<td>Left View of Real Image after Rendered using 3D Cloud Points</td>
<td>62</td>
</tr>
<tr>
<td>5.19</td>
<td>Reconstructed geometry of real dataset (Left View)</td>
<td>63</td>
</tr>
<tr>
<td>5.20</td>
<td>Reconstructed geometry of real dataset (Right View)</td>
<td>63</td>
</tr>
<tr>
<td>5.21</td>
<td>Graph showing how the Diameter varies along the Length</td>
<td>64</td>
</tr>
</tbody>
</table>
LIST OF TABLES

<table>
<thead>
<tr>
<th>Table</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.1 Results of Applying Gaussian Filter using our method</td>
<td>31</td>
</tr>
<tr>
<td>4.2 Results of Applying Threshold Filter using our method</td>
<td>33</td>
</tr>
</tbody>
</table>
ACKNOWLEDGEMENTS

I would like to thank my advisor, Dr. Thomas Wischgoll, for his guidance and input on the thesis development process. I also wish to thank Dr. Arthur Goshtasby and Dr. Thomas Hartrum for serving as members of my MS thesis defense committee, giving the constructive criticism necessary to produce a quality technical research document. I would also like to thank the Department of Computer Science and Engineering and Dr. Thomas Sudkamp, the Department Chair, for giving me the opportunity to obtain my degree at Wright State University.
1. INTRODUCTION

1.1 Motivation

1.1.1 Analyzing Very Large Scale Data

Data, in any area, can be defined as the collection of meaningful information from which some conclusion can be drawn. It is the representation of facts, concepts in a formalized manner suitable for interpretation or processing by humans or by any machine such as computers. In computing industry, data can be defined as the information that can be moved or processed. In brief, it can be explained as information transformed into digital form. In the field of computer graphics, data can be turned into image files.

Due to the tremendous growth in the field of science, the amount of data being captured is significantly increasing and due to which the scope for analyzing such data has arisen. Particularly, in the field of medical imaging, the quantity and quality of the data being captured is significantly higher. Medical imaging is a field of science which involves the usage of techniques and processes to create the images of the human body (or parts thereof) for clinical purposes.

Analysis of image data plays an important role in understanding and solving the scientific problem at hand. By analyzing, we can extract features which are of interest to us leaving the rest behind so that it can be used efficiently for further analysis. The size of these medical images can vary from few kilo-bytes to several tera-bytes which makes the
analyzing process increasingly difficult. For example, a CT-scanned human heart dataset of a size of 7.8 giga bytes can not be analyzed on off-the-shelf desktop computers equipped with standard configurations. Such an analysis is possible on super computers which are very expensive and not easily accessible to researchers. Hence, there is a need for some technique which it is capable of performing the analysis on off-shelf desktop computers.

1.1.2 3D Reconstruction Using Calibrated Images

Scene reconstruction from uncalibrated images has been an important research area in the field of computer vision. Researchers [34] are able to achieve the 3D reconstruction for a pair of stereo images using epipolar constraint. Epipolar constraint [27], is that for any given point x on image 1, its corresponding point x’ must lie on the epipolar line of x in image 2. This reduces the search of the corresponding point from 2 dimensions to 1 dimension. This type of classical method would require careful selection of the matching features and most of the cases this approach is used with uncalibrated images which is error sensitive process. Hence we need a method using which we can bypass the process of manually finding the matching points in a pair of stereo images. This would enable us to automatically reconstruct the 3D coordinates of a number of points in a scene given a pair of images obtained by cameras of known relative positions and orientations.
1.2 Thesis Objective

The thesis’s work mainly consists of two parts. In the first part, we present a method for the analysis of very large scale volumetric data on off-the-shelf desktop computers making optimal use of the available memory. So, the objective of this part of the thesis work was to analyze the large scale volumetric image data and evaluate the performance of the proposed algorithm over the other trivial methods on off-the-shelf desktop computers with standard configurations.

In the second part, efforts have been made to develop an algorithm to automate the reconstruction of 3D geometric model of computer generated phantom wire dataset using the basic understanding of epipolar concept. The latter work is composed of the previous work.

The main objectives of this thesis work are summarized as follows:

1. Present a method developed to work with very large scale volumetric datasets on the off-the-shelf common desktop with standard configurations by using main memory as cache and hard disk as the extended memory.

2. Gaussian and threshold filters were implemented to show the time performance of the proposed method against the trivial method to support our arguments.

3. Implemented filters were used to remove the air bubbles from the pig heart blood vessels deposited during the process of scanning.
4. Proposed method was used to map the bi-planar phantom wire dataset thereby validating the algorithm presented for the reconstruction 3D of calibrated images.

1.3 Thesis Organization

Chapter 2 presents some of the basic concepts such as introduction to 2D and 3D graphics and 3D data definition. Then a brief introduction has been given to computer vision and its relationship with various other fields. Finally, a detailed description about some of the advanced imaging techniques is given. In Chapter 3, a brief description of related works in the area of out-of-core technique and 3d reconstruction are given. In Chapter 4, a detailed description of analysis of very large scale volumetric image data over an ordinary computer with standard configuration is given. Then a complete description of mmap() function, its parameters description in the form of table and its return values is given. Subsequently, the implementation for Gaussian and threshold filters are given, followed by the results. In Chapter 5, Epipolar geometry is explained in detail with the explanation of fundamental matrix. Then the proposed algorithm and its results are explained in detail with the specifics like 3D intersection point algorithm and normal calculations. Finally, Chapter 6 presents a brief summary of the thesis and future works.
2. GENERAL CONCEPTS

2.1 Image Processing

Research in computer graphics covers a broad range of topics including both photo realistic and non photo realistic image synthesis, image-based modeling and rendering, applications of wavelets and other multi-resolution methods, curve and surface design, range scanning, surface reconstruction and modeling, motion capture, motion editing, physics-based modeling, animation, interactive 3D user interfaces, image editing and color reproduction. The project is mainly focused on processing of large scale image data. Image processing, a technical discipline, in the area of computer science, is a technique in which data from an image are digitized and various mathematical operations are applied to data. It is basically done to extract some of the useful feature information in other words revealing the visual content. The relation between image processing, computer vision and various other fields are as shown in the figure [39] below.
2.2 General Definition of 3D Dataset

3D computer graphics are often referred to as 3D models. The model is contained within the graphical data file called dataset. A 3D model is the geometric representation of any object in three dimensional spaces. The geometric model by itself is technically not graphics unless it is visualized as two dimensional images through a process called 3D rendering, or used in non-graphical computer simulations and calculations. The process of creating 3D model can be divided into 3 basic steps namely 3D modeling
which describes the shape of the model, layout and animation which describes position of the object in the image scene and 3D rendering which produces the image of an object.

As a part of this thesis work, emphasis has been given on the first stage of the 3D computer graphics that is 3D modeling which describes the geometric model of the data. The two main sources of the 3D model are those created by an artist on a computer or by using some kind of 3D modeling tool and those scanned into computer from real-world objects. The type of data used for this thesis work is obtained by the second method. A 3D scanner is a device that analyzes a real-world object or environment to collect data on its shape and possibly its appearance (i.e. color). The collected data can then be used to construct digital, three dimensional models useful for a wide variety of applications.

Due to the recent developments in the imaging technologies, the scanners, such as lasers which retrieve a surface representation or CT or MRI which generate volumetric images, have become very sophisticated in terms of the details while capturing the information. With these devices continuing to improve in resolution, the resulting amount of data increases dramatically as well.

The purpose of a 3D surface scanner is usually to create a point cloud of geometric samples on the surface of the subject; if color information is collected at each point, then the colors on the surface of the subject can also be determined. The point clouds produced by 3D scanners are usually not used directly. Most applications do not use point clouds, but instead use polygonal 3D models. The process of converting a point cloud into a polygonal 3D model is called reconstruction. Reconstruction involves finding and connecting adjacent points in order to create a continuous surface.
Most of the times, a single view of an image is not sufficient to obtain information about all sides of the subject, rather multiple views from different directions are required. After this information is captured it is processed for further analysis and studies which can be referred to as image processing. Image processing is a form of information processing for which the input is an image such as an image file, data file etc; the output is not necessarily the image but can be a set of features of the image which is of interest to the user or which can be used for further analysis.

2.3 A Brief Overview of Computer Vision

Computer vision, a scientific discipline, is a technology concerned with computational understanding and use of the information present in visual images. Images can be of any form, such as views from multiple cameras, multi-dimensional data from a medical scanner, or even video sequences. It can also be described as the compliment of the biological vision. In biological vision, the visual perception of humans and various animals are studied, resulting in models of how these systems operate in terms of physiological processes. Computer vision, on the other hand, studies and describes artificial vision systems that are implemented in software and/or hardware. However, the goal of computer vision is primarily to enable engineering systems to model.

It begins with the acquisition of images. A camera produces a grid of samples of the light received from different directions in the scene. The position within the grid where a scene point is imaged is determined by the perspective transformation.

One central objective of image interpretation is to infer the three-dimensional (3D) structure of the scene from images that are only two-dimensional (2D). As a part of this
thesis work, an attempt has been made to use the basic concept of computer vision i.e., automatic 3D reconstruction of phantom wire data-set using 2 different camera views of the same object taken at an angle of 90 degrees each.

2.4 Some Imaging technologies

Medical imaging is a field where they use some technologies such as x rays, gamma-rays, radio waves, magnetic field and other techniques to produce images of organs and other internal structure of the human body. The purpose is to detect and diagnose diseases. In general, these imaging technologies are computationally intensive and take advantage of a wide diversity of fundamental physical interactions.

2.4.1 X Rays

Diagnostics x rays are one of the powerful imaging tools available. X-rays are a type of radiation used in imaging that uses short wavelength (electromagnetic waves) energy beams capable of penetrating most substances except for heavy metals. X-rays help diagnose many types of cancers, including sarcomas, lymphomas, and lung cancers [39]. X- rays allow the physician to visualize certain internal body conditions with little or no invasive procedures. For proper diagnosis, attention has to be given to patients positioning and shielding.
X-rays, invisible to the naked eye, are directed at the part being examined which passes through the patient's body. Various structures absorb the x-ray photons differentially: bones more than soft tissues; other organs and tissues such as muscle producing shadows of varying intensity. The image is recorded by a detection device, either a fluorescent screen (screening) or photographic film (radiography). However, using x-rays alone it is not possible to distinguish between soft tissues of the same density. In order to accomplish this, various liquid or gaseous contrast media are used [39]. The last of the process that is filling liquids or contrast media requires careful attention; otherwise lots of false information is added.

2.4.2 CT (Computed Tomography)

As the technology advances, the imaging of the human body has seen substantial change. For example, using x-rays have their limitations such as there is no clue about the depth, size or even shape. On the other hand, a new form of x-rays-based technology called CT (computed tomography) can detect all these.

A narrow x-ray beam traverses the body in an axial plane (the anatomical term for a vertical plane dividing the body from front to back) and multiple detectors, surrounding the body, record the strength of the exiting x-ray. This data is analyzed, integrated, and reconstructed by a computer to produce images of cross-sections or ‘slices’ of an organ or region of the body [39]. Using sophisticated software, one can visualize the 3-dimensional structure of the specific organs.
2.4.3 MRI (Magnetic Resonance Imaging)

MRI (Magnetic Resonance Imaging, also called NMR for Nuclear Magnetic Resonance), an imaging technology, is based on the observation that the nucleus of the hydrogen atom (a proton) acts like a small bar magnet which, when it is immersed in a magnetic field, resonates at a frequency proportional to the strength of that field. If a specimen is immersed in a magnetic field that varies in strength in different regions of space, the protons at different positions will resonate at different frequencies (typically radio frequencies) -- the strength of the resonance at a particular frequency therefore measures the number of protons in a particular region of space. A complete map of the spatial distribution of the protons can be built up by varying the shape of the magnetic field over time and repeatedly measuring the strength of the resonance at different frequencies [25, 26].

MRI is routinely used to image various parts of human beings -- the heart and brain are particularly popular. The cover of Nature was graced by an MRI micrograph of a single living cell at a resolution of 10 microns [23] -- what makes this particularly interesting is that no theoretical resolution limit for this technology is yet in sight, and the practical limit to resolution is still unclear. The recent dramatic advances in superconductors [24] should provide magnets that are more powerful and much cheaper than exist today, which will make MRI microscopy cheaper and improve its resolution. Further progress is expected with that the size of the captured data will also increase dramatically.
3. RELATED WORK

As the medical imaging technologies are improving, the size of the data captured is also increasing dramatically. Analyzing such large scale data and reconstructing the 3D scene using calibrated images are active research areas in the medical image processing community. A lot of research work has been done in the field of medical image processing and 3D scene reconstruction. The area of focus in this work has been to process the large scale data on off-the-shelf desktop computers with limited memory and also to automate the process of 3D reconstruction of real human hearts using calibrated images. An out-of-Core technique has been proposed in this project. So the related works are in the field of out-of-core technique and 3D reconstruction using epipolar geometry. Few of the references given below are suited for interactive use and the development of multi-resolution approaches that would allow for scalable analyzing techniques.

Cox and Ellsworth [1] propose a framework for out-of-core scientific analysis for visualization systems based on application-controlled demand paging. The paper is built on the fact that many important visualization tasks only need to touch a small portion of large datasets at a time. Thus, the idea is to just page in the necessary data on demand. Unfortunately, as the paper explains, the operating system paging sub-system is not effective for such visualization tasks, and leads to inferior performance.
Ueng et al. [2] present a related approach. In their work they focus on computing streamlines of large unstructured grids, and they use an approach quite similar to Cox and Ellsworth’s, where the idea is to perform on-demand loading of the data necessary to compute a given streamline. Instead of changing the way the operating system handles the I/O, the authors modified the organization of the actual data on disk, thereby implementing optimized software for the task.

Yu Goa, Deng and Wu [3] presented a new method for efficient view-dependent out-of-core technique for large-scale and complex scenes. They had two parts, one consists of the out-of-core preprocessing and the other one of the run-time view-dependent rendering. In the preprocessing phase, they divided the scenes to be rendered into small blocks and built a bounding box hierarchy. Then, they constructed a continuous hierarchy level of detail for each hierarchical node using a dynamic mesh simplification method which supported topological structure modification. At run time, they used multi-threaded techniques to overlap rendering and pre-fetching. The rendering thread traversed the hierarchy and rendered the scene by view-dependent selective refinement. The pre-fetching thread predicted the motion of the viewer and pre-fetched the data that would be seen next. The idea was to keep the data ready for viewing so that rendering could be done continuously.

Farias and Silva [7] present a set of techniques for direct volume rendering of arbitrarily large unstructured grids on machines with limited memory. One of the techniques described in the paper is a memory-insensitive approach which works by traversing each cell in the dataset, one at a time, sampling its ray span (in general a ray would intersect a convex cell twice) and saving two fragment entries per cell and pixel
covered. The algorithm then performs an external sort using the pixel as the primary key, and the depth of the fragment as the secondary key, which leads to the correct ray order that exactly captures the information necessary to perform the rendering.

An out-of-core iso-surface technique was first given by Chiang and Silva [8]. The authors use the I/O-optimal interval tree [13] to solve the interval search problem. An interesting part of their work is that they were able to visualize large datasets exceeding the available memory efficiently and most of the preprocessing part was done on a machine with limited memory. Though this technique is quite fast in terms of actually computing the iso-surfaces, the associated disk and preprocessing overhead is substantial.

Later, Chiang et al. [20] further improved (i.e., reduced) the disk space overhead and the preprocessing time, at the cost of slightly increasing the iso-surface query time, by developing a two-level indexing scheme, the meta-cell technique, and the BBIO tree which is used to index the meta-cells. A meta-cell is simply a collection of contiguous cells, which is saved together for fast disk access.

Along the same lines, Sulatycke et al. [9] describe an out-of-core isosurface extraction algorithm. They propose a scheme that simply adapts the in-core data structure to an out-of-core setting. In order to reducing the drawbacks of disk input/output overhead, they describe a multi-threaded implementation by overlapping computation with I/O. Basically, the authors have an I/O thread that reads the active cells from disk, and several iso-surface computation threads. They have also given experimental results for small regular grids in their paper.
Uytterhoeven [11] in his PhD thesis work presented a wavelet-based scheme to process very large images that do not fit in the memory of a single computer. Such images were divided into blocks. Because of the spatial locality properties of wavelets these blocks could be processed independently, allowing efficient parallel programming.

Hastings et al. [10] presented a method to work with large scale medical image data. They used the power of PC clusters built from low-cost, commodity components which are widely used, and resulting in a change in the way hardware can be scaled and leveraged in a cost-effective manner. The authors also extended their work by addressing challenging issue of data storage and retrieval, work flow management and also demonstrated the XML based meta-data along with their distributed execution environment.

Masciari et al. [36] describe the implementation of an out-of-core technique for data analysis of very large data sets with a sequential and parallel version of the clustering algorithm ‘AutoClass’. Authors discuss the out-of-core technique and show the performance results in terms of execution time.

Despina Kontos et al. [16] propose an approach based on a supervised framework for reducing the dimensionality of the feature space when characterizing and classifying spatial Regions of Interest (ROIs). Their approach uses the statistical techniques of Bootstrapping simulation, Bayesian Inference and Markov Chain Monte Carlo (MCMC), to select the most informative features according to their discriminative power across distinct classes of data. This reduces the dimensionality of the initial feature space and also improves the classification of the ROIs, since features providing irrelevant
information with respect to class membership are discarded. They also introduced a weighted Euclidean Distance designed to effectively classify the ROIs. They have also demonstrated its effectiveness in classification experiments (using established classifiers) and in similarity searches and showed the effect of reducing the dimensionality of the initial feature space.

For the analysis of multi-camera imagery, Feldmann et al. [35] present a new concept for the extension of epipolar image analysis to more general camera configurations like circular camera movements usually occurring for concentric mosaic acquisition. The authors present a method for 3D scene depth reconstruction which is not restricted to horizontal, linear, and equidistant camera movements. Their algorithm uses all available views of an image sequence simultaneously. Instead of searching for straight lines in the epipolar image they compute the trajectories of particular points through the image cube. The best match is assumed to correspond to the true depth.
4. ANALYSIS OF LARGE SCALE IMAGE DATA USING OUT-OF-CORE TECHNIQUE

4.1 Image Processing

Image processing is an active area of research in diverse fields; such as medicine, astronomy, microscopy, seismology, defense, industrial quality control, and the publication and entertainment industries. Image processing is a technique in which data from an image is processed to extract some feature information. One of the first steps is to enhance an image so that it becomes more useful to perform some of the interpretation and recognition tasks. A 2D image is usually interpreted as a two-dimensional array of brightness values which can be processed digitally using a computer. Each number representing an intensity value of the image at a particular location is called a pixel. For example, a typically digitized image may have 512 * 512 or 262,144 pixel, although much bigger images are available.

As it has been mentioned earlier, the improvement in the area of devising technology has been tremendous with the invention of some sophisticated technologies like MRI (Magnetic Resonance Imaging), CT (Computed Tomography), Photoelectron imaging etc., and consequently the size of the data being captured has also increased. Computers are indispensable for the analysis of large amounts of data, for tasks that require complex computation, or for the extraction of quantitative information.
Image processing is an important stage of the research process. Analysis of image data plays an important role in understanding and solving the scientific problem at hand. By analyzing an image data, we can extract features only which are of interest, leaving the rest behind, so that it can be used efficiently for further analysis. As the first step, the image data has to be analyzed, so that the underlying structure of the data can be understood for further processing, such as visualization, which is a boon to the simulation researchers, since it delivers a clear image of what is going on in a simulated phenomenon.

But the analysis of image data is hindered by the increasing size of data. In the medical imaging field, due to the advanced development of scanning technology with higher resolution, driven by the demand for more accurate results, the size of data captured during the process is enormous which makes the analysis of data nearly impossible on an ordinary computer with standard configuration. These advanced image acquisition devices make it possible for biomedical researchers to capture the structural and functional information at a very high resolution. With very high end scanning, it is possible to generate images up to 20 gigabytes which exceeds the available memory in most computers.
4.2 Analysis of Large Scale Image Data

After acquiring the raw volume data using any imaging technique, the next step is to analyze it and the reason is the presumption that the underlying structure yields valuable information, otherwise unobtainable. However, most existing algorithms do not scale well with the growing data size and other parts of the data analysis pipeline. It is understood that, the amount of data collected and stored electronically is being doubled every three years [16]. Several techniques have been proposed by researchers to retrieve the required information from these kinds of large datasets, not only related to medical field but also from many different areas of interest like satellite data processing, microscopic data, mining intersecting patterns[6]. Hence, the immediate concern or the emerging problem is to interpret the information available. The essential problem is that the data volumes are overwhelming the existing analysis tools. There are super computers such as with the DOE's ASCI Program with huge memory and multiple processors to work with these large scale datasets but the idea is to effectively manage and handle these large scale datasets on a regular PC with standard configurations as these supercomputers are very expensive and are hardly available to researchers all the time. Hence, we need a technique to maintain and analyze large scale data more efficiently on a regular machine. Our approach to solve this problem includes some operating system API’s which is explained in the next section.

As a part of this thesis work, an out-of-core technique has been proposed which can handle any sized data on an ordinary computer. Out-of-Core functionality is achieved by implementing memory mapping function. The details of out-of-core technique are
given in the next section. Using memory mapping method, there is no limit on the size of the data that can be processed. For example, a volumetric image data of a human heart with a resolution of 5μm results in a dataset of 4.7 gigabytes which can be mapped on to a process’s address space using memory mapping and analyzed on an ordinary computer with limited memory space. This approach makes efficient use of the available main memory and hard disks. The details are given in the following sections.

4.3 Out-of-Core Technique

Out-of-core techniques are designed and developed to handle very large scale data on common, off-the-shelf desktop computers and to make efficient use of the system’s main memory. The out-of-core method avoids bringing all the data structures into the main memory at once. Instead it loads only parts of the data into the main memory at that time, thereby making efficient use of the data locality of the filtering algorithms. In applications requiring out-of-core techniques, the algorithm must process data that is too large to fit into the system’s main memory at one time. For large scale data, one needs to have large system memory which is expensive. Often times the system does not even support the amount of memory required. Our approach overcomes this problem by making use of the hard disk as an extended memory and main memory as cache. Researchers have proposed many such external algorithms to overcome the system’s main memory problem [6, 14, 36] while dealing with large scale data. The following section explains the algorithm based on memory mapping implemented for this project to achieve out-of-core functionality.
4.4 Memory Mapping

Memory-mapping is a concept where a file is a segment of virtual memory which has been assigned a direct byte-for-byte correlation with some portion of a file. This resource or file can be referred to through a file descriptor by the operating system. Once the mapping is done, this correlation between the file and the memory space permits applications to treat the mapped portion of the file as if it were primary memory. In traditional file I/O involving read, data is copied from the disk to a kernel buffer, and then the kernel buffer is copied into the process's heap space for use. In memory-mapped file I/O, data is copied from the disk straight into the process's address space, into the segment where the file is mapped. The file is then accessed by references to memory locations with pointers. This way, I/O on the file is done without the overhead of handling the data twice.

4.4.1 Memory Mapping Function Description

Following is a piece of code from the project showing the usage of `mmap()` function,

```c
int realfd, backingfd;          // file descriptors
unsigned char *realfile;
unsigned char *backingfile;
```
// opening the file for read only purpose

if ((realfd = open(argv[1], O_RDONLY)) < 0)
{
    cout<<"Backing store: couldn't open the real file\n"<<endl;
    return 0;
}

// opening the file for both read and write

if ((backingfd = open(argv[2], O_CREAT | O_RDWR)) < 0)
{
    printf("Backing store: couldn't open the backing file\n");
    close(realfd);
    return 0;
}

if(ftruncate(backingfd, (long)width*depth*height) < 0)
{
    //error(0,errno,"fail\n");
    printf("Backing store: couldn't allocate space for backing data\n");
    close(realfd);
    close(backingfd);
    return 0;
}
//reading a file

realfile = (unsigned char *) mmap (NULL, 
(long)width*depth*height, PROT_READ, MAP_SHARED, realfd, 0);

//write into a file

backingfile = (unsigned char *) mmap (NULL, 
(long)width*depth*height, PROT_WRITE, MAP_SHARED, 
backingfd, 0);

if(backingfile == MAP_FAILED)
{
    printf("couln't mmap the backing file\n");
    close(realfd);
    close(backingfd);
    return 0;
}

if(realfile == MAP_FAILED)
{
    cout<<"map error"<<endl;
    close(realfd);
    return 0;
}
As we can see above, mmap() function is called twice. Once while reading the file and the next time while writing into the file.

The format of mmap() function call is as follows,

```c
void *mmap(void *addr, size_t len, int prot, int flags, int fildes, off_t off);
```

The first parameter of mmap() function is implementation dependent. It specifies the address in the process address space from where the mapping should begin. If this parameter is explicitly given, then the address range starts at address and continues up to len byte which is the second parameter. Ideally, it is best to let system choose the address by specifying NULL as the value for the argument addr and not to specify MAP_FIXED (explained later).

The second parameter len specifies the length of address space required for mapping. Usually, it will be equal to the length of file being mapped. In our case, the length of the file is equal to width*depth*height.

The third parameter prot specifies the protection mode. This parameter determines whether read, write, execute, or some combination of accesses are permitted to data being mapped. In our case, since we read and also write using mmap() function, this parameter changes accordingly. The options available for this parameter are PROT_READ, PROT_WRITE, PROT_EXEC, PROT_NONE or any combination of these flags.
The fourth parameter *flags* provide other information about the handling of the mapped data. The value of flags can be MAP_SHARED, MAP_PRIVATE and MAP_FIXED. MAP_SHARED means changes are shared between processes. MAP_PRIVATE means changes are private to the calling process only. MAP_FIXED means the value of *address* must be *addr*, otherwise mmap() will return failure. If a MAP_FIXED request is successful, the mapping established by mmap() replaces any previous mappings for the process' pages in the range \([address, address + len]\).

The fifth parameter *fildes*, specifies the file descriptor for the opened file. The *mmap()* function adds an extra reference to the file associated with the file descriptor *fildes* which is not removed by a subsequent *close()* on that file descriptor. This reference is removed when there are no more mappings to the file.

And the last parameter *off* specifies the offset. In this case, if *off* is non-zero, the behavior of *mmap()* is undefined. If *fildes* refers to a valid memory object that is not accessible from the calling process then *mmap()* function fails.

### 4.5 Advantages of Memory Mapping

Memory-mapped files offer a unique memory management feature that allows applications to access files on disk in the same way they access dynamic memory—through pointers. With this capability it can map a view of all or part of a file on disk to a specific range of addresses within the process's address space. And once that is done, accessing the content of a memory-mapped file is as simple as dereferencing a pointer in the designated range of addresses. So, writing data to a file can be as simple as assigning a value to a de-referenced pointer.
The main purpose of memory mapping a file is to increase the performance, especially when dealing with large scale datasets. Memory mapping is based on the concept of a virtual memory manager which also takes care of paging the file. The size of the page is typically decided by the operating system for optimum performance. Since page file management is one of the most critical elements of a virtual memory system, loading page sized sections of a file into physical memory is typically a very highly optimized system function. Note that only parts of the file are loaded into the main memory when the algorithm uses the corresponding pointer area within the file. This may result in other areas of the file being removed from the main memory. Hence, the operating system automatically takes care of paging the file.

One other important benefit of memory-mapped file is the automatic management of large files as smaller chunks. When dealing with files that are significantly bigger than the system's total memory size, trying to load the entire on-disk contents into memory can be a serious set back to the performance as the operating system reads from disk into memory and simultaneously pages from memory back to disk. By memory-mapping the large file, not only is the page file completely bypassed, but also the system only has to load the smaller page-sized sections being read or written at any given time, and the rest is taken care of by the operating system.

Oyetunde Fadele[27] has done a performance evaluation between traditional read/write/lseek system calls with the memory mapping. He has shown that file I/O performs better with memory mapping than with traditional read/write/lseek system calls. Results from the sample application in his work shows that the read/scale/free calls consistently takes as much as 103 percent more time than the equivalent mmap/scale/free
alternative. In his study, he shows that the read time is 15 percent more than the mmap time.

4.6 Limitations of Memory Mapping

Since memory mapped pages can be stored back to their file when physical memory is low, it is possible to mmap files with orders of magnitude larger than both the physical memory and swap space. The only limit is the address space. The theoretical limit is 4 GB on a 32-bit machine - however, the actual limit may be smaller since some areas will be reserved for other purposes. In this project, all the experiments were conducted on 64-bit machines where the limit is significantly higher (4 TB).

4.7 Implementation and Experimental Results

Since the memory mapping function can be used to map very large scale data on to system’s limited memory, we propose the idea of using memory mapping to achieve the out-of-core functionality. All the experiments were conducted on an ordinary PC with 3.2GHz Intel Pentium-IV CPU, 2GB of main memory and a NVIDIA GeForce 6600 graphics card with 256 mega bytes of video memory running on Mandriva 2007. Memory mapping function is also available in windows and other operating systems.

A pig heart dataset of size 768 megabytes was mapped using mmap() function. The purpose of this work was to remove air bubbles introduced during the process of injecting the contrast agent into the organs. These air bubbles cause problems in the later analysis process.
Gaussian filter, a noise smoothing method, was implemented to remove the air bubbles. It uses a normal distribution for calculating the transformation to apply to each pixel in the image. The equation of Gaussian distribution in N dimensions is

\[ G(r) = \frac{1}{(2\pi \sigma^2)^{N/2}} e^{-r^2/(2\sigma^2)} \]

or specifically in three-dimensions,

\[ G(x, y, z) = \frac{1}{(2\pi)^{3/2} \sigma^3} e^{-\left(\frac{x^2 + y^2 + z^2}{2\sigma^2}\right)} \]

Where \( r \) is the blur radius \( (r^2 = x^2 + y^2 + z^2) \) and \( \sigma \) is the standard deviation of the Gaussian distribution. Each pixel's value is set to a weighted average of that pixel's neighborhood. The original pixel's value receives the heaviest weight (having the highest Gaussian value), and neighboring pixels weight decreases symmetrically as distance from the center increases. This results in a blur that preserves boundaries, edges better than other and also removes the air bubble when used based on the threshold value. However, the computational task becomes more and more expensive as the size of the dataset increases, as it requires performing operation on a per point basis.
Fig 4.1 Showing pig heart with air bubbles

Fig 4.2 Showing pig heart with the air bubbles removed
As discussed earlier, one of the objectives of this work is to use ordinary computers with limited memory to analyze very large scale volumetric data. In order to test the functionality of memory mapping method, Gaussian filter was also applied on various different volumetric datasets. Most of these datasets are of real human heart taken at different resolutions and hence with varying sizes. Some of the datasets used in the experiment are of sizes 768 MB, 1.7GB, 3.3 GB and 4.7 GB.

Similar to Gaussian filter, threshold filter is also applied to remove noise in the dataset. The threshold value acts like a mask; it filters out or replaces the pixel’s intensity value with its own value if it is below the given threshold, otherwise it retains the original value.

All these datasets were mapped using the `mmap()` function call and the times elapsed to process the data using Gaussian and threshold filters were noted (table 4.1 and table 4.2). Also, the same datasets were used to apply ITK’s (Insight Segmentation and Toolkit) Gaussian and threshold filters. ITK filters along with VTK’s pipeline execution concept works well until the dataset size exceeds a certain limit. Once the dataset size exceeds the limit, it fails to load them due to limited available main memory. In order to compare our proposed method, a time performance analysis was done and results are as shown in fig 4.3.
Results

As discussed above, except for the pig’s heart dataset, most of the datasets used for this experiment are of the human’s heart, with different sizes and resolutions. All the datasets are of 8 bit-depth and are three dimensional. Average time taken after several runs is given in the table below.

For the Gaussian Filter, the following timings were determined.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Size</th>
<th>Dimensions</th>
<th>Average Time Taken</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pig Heart</td>
<td>768 Mega bytes</td>
<td>815 * 795 * 1265</td>
<td>12 Minutes 8 Seconds</td>
</tr>
<tr>
<td>Human Heart</td>
<td>1.7 Giga bytes</td>
<td>768 * 616 * 3903</td>
<td>29 Minutes 38 Seconds</td>
</tr>
<tr>
<td>Human Heart</td>
<td>3.3 Giga bytes</td>
<td>902 * 960 * 3873</td>
<td>50 Minutes 20 Seconds</td>
</tr>
<tr>
<td>Human Heart</td>
<td>4.7 Giga bytes</td>
<td>1068 * 972 * 4426</td>
<td>1 Hr 07 Minutes 25 Seconds</td>
</tr>
</tbody>
</table>

Table 4.1 Results of applying Gaussian filter using our proposed method

The values used to plot the graph shown below are the results of applying the Gaussian filter using our proposed method.
Fig 4.3 Graph showing the time performance with the increase in data size for Gaussian filter
For Threshold Filter, the timings were as follows.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Size</th>
<th>Dimensions</th>
<th>Average Time Taken</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pig Heart</td>
<td>768 Mega bytes</td>
<td>815 * 795 * 1265</td>
<td>1 Minute 8.55 Seconds</td>
</tr>
<tr>
<td>Human Heart</td>
<td>1.7 Giga bytes</td>
<td>768 * 616 * 3903</td>
<td>3 Minutes 39.57 Seconds</td>
</tr>
<tr>
<td>Human Heart</td>
<td>3.3 Giga bytes</td>
<td>902 * 960 * 3873</td>
<td>5 Minutes 43.87 Seconds</td>
</tr>
<tr>
<td>Human Heart</td>
<td>4.7 Giga bytes</td>
<td>1068 * 972 * 4426</td>
<td>8 Minutes 35.88 Seconds</td>
</tr>
</tbody>
</table>

Table 4.2 Results of applying Gaussian filter using our proposed method
Fig 4.4 Graph showing the time performance with the increase in data size using threshold filter

The time performance results for the pig’s heart dataset, which works on both our proposed method and ITK filters, is given below,
Fig 4.5 Time comparison chart between ITK’s Gaussian filter and our method

Fig 4.6 Time comparison chart between ITK’s threshold filter and our method
For a small dataset like the Pig’s Heart (768 MB), it can be seen from the above that the shortest time taken by ITK’s Gaussian filter was 42 Minutes whereas the method proposed as a part of this work just took an average of 12 Minutes and 8 seconds. The largest dataset with which we were able to work using the ITK filter was of 768 MB, but the processing of the data took more time than the proposed method’s processing time. In this project, it has been shown that, ITK filter libraries are limited with respect to the size of the dataset they can handle whereas when using the algorithm implemented for this project, there is virtually no limit on the size of the dataset. The data set size that can be processed is only limited by the size of the hard disk and the process’s address space. Hence, the novelty of the described technique is the ability to apply the implemented filters to data sets of almost any size using common, off-the-shelf desktop computers.
5. 3D RECONSTRUCTION USING CALIBRATED IMAGES

The process of 3D reconstruction of images in the world coordinates based on 2D images is a complex problem and there are many methods which require fine tuning to adapt to the scene and improve reconstruction. Reconstruction of view-independent 3D information from a set of static images has drawn the attention of researchers for some years now [27, 28]. The approaches are different with different applications. If a pair of entities from two distinct images correspond, and the camera position, orientation and intrinsic parameters of the two images are known (or estimated), it is possible to compute the position of those entities in 3D space. This position can be expressed directly in 3D referential or as a depth—the distance relative to a reference camera. However, each of these entity types has some type of ambiguity associated such that, given two entities in different images, it is not straightforward to determine whether they corresponded or not [29].

5.1 Epipolar Geometry

Epipolar geometry is one of the common methods in the field of computer vision used to reconstruct 3d image from a set of two 2D images. The epipolar geometry is the intrinsic projective geometry between two views. It is
independent of scene structure, and only depends on the cameras' internal parameters and relative position.

There are two ways of extracting three-dimensional structure from a pair of images. In the first and classical method, known as the calibrated route, we firstly need to calibrate both cameras (and viewpoints) with respect to some world coordinate system, calculate the so-called epipolar geometry by extracting the essential matrix of the system, and from this compute the three-dimensional Euclidean structure of the imaged scene.

However it is the second or uncalibrated route, which more likely corresponds to the way in which scanning systems in biology and medicine determine three-dimensional structure from vision. In an uncalibrated system, a quantity known as the fundamental matrix is calculated from image correspondences, and this is then used to determine the projective three-dimensional structure of the imaged scene.

In both approaches the underlying principle of binocular vision is that of triangulation. Given a single image, the three-dimensional location of any visible object point must lie on the straight line that passes through the centre of projection and the image of the object point (see figure 5.1 [38]). The determination of the intersection of two such lines generated from two independent images is called triangulation.
Figure 5.1 Concept of triangulation in stereo images

In the above figure we can see, that the determination of the scene position of an object point through triangulation depends upon matching the image location of the object point in one image to the location of the same object point in the other image. The process of establishing such matches between points in a pair of images is called correspondence, which relates pixels in image 1 to pixels in image 2. In order to determine the pixel in image 1 corresponding to a pixel in image 2, a search maximizing the similarity criterion must be performed over some region of image 2. Such regions reduce to lines (epipolar lines) and if the fundamental matrix (described later) is known, the correspondence problem is reduced from 2D search to a 1D search problem.
The fundamental matrix is the algebraic representation of epipolar geometry. In the following we derive the fundamental matrix from the mapping between a point and its epipolar line, and then specify the properties of the matrix. Given a pair of images, it can be seen in figure 5.2 that for every point \( x \) in one image, there exist a corresponding epipolar line \( l' \) in the other image. Any point \( x' \) in the second image matching the point \( x \) must lie on the epipolar line \( l' \). The epipolar line is the projection in the second image of the ray from a point \( x \) through the camera centre \( C \) of the first camera. Thus, there is a map

\[
x \rightarrow l'
\]

from a point in one image to its corresponding epipolar line in the other image. It will turn out that this mapping is a (singular) correlation that is a projective mapping from points to lines, which is represented by a matrix \( F \), the fundamental matrix.
The epipole is the point of intersection of the line joining the optical centers that is the *baseline*, with the image plane. Thus, the epipole is the image, in one camera, of the optical center of the other camera.

The epipolar plane is the plane defined by a 3D point $M$ and the optical centers $C$ and $C'$ while the epipolar line is the straight line of intersection of the epipolar plane with the image plane. It is the image in one camera of a ray through the optical center and image point in the other camera. All epipolar lines intersect at the epipole.

As described above, to get the epipolar lines from arbitrary points in the other image, the epipolar geometry has to be determined and it is described by $3 \times 3$ matrix called fundamental matrix. Fundamental matrix has only 7 degrees of freedom. Therefore fundamental matrix can be solved with 8 corresponding
points in the images. Finding the corresponding points in the images is a tedious, complex and time consuming process [30].

As part of this thesis work, effort has been made to improve the process of reconstruction of 3D image using the same basic concept of epipolar geometry but by bypassing the process of calculating the fundamental matrix which is as explained earlier time consuming process and it requires manual intervention to locate the 8 corresponding points which is once again time consuming and tedious. However, if the geometric configuration of the cameras and image planes are known, this step can be skipped. Hence, we developed an algorithm which does find the 3D intersection point given the camera information and the two views of the images taken orthogonally to each other.

5.2 Intersection of 2 Lines in 3D

After knowing the camera and object location in 3d, the next step is to implement the basic understanding of ray casting i.e., finding the intersection of two lines joining the respective camera eye and the object scene in 3d. In order to find the intersection, first we need to find the shortest distance between the 2 lines. This idea has been proposed by Bourke [38] and is as follows,

5.2.1 Algorithm

Generally two lines in 3 dimensions don't intersect at a point, they may be either parallel (no intersections) or they may be coincident (infinite intersections) but most often only their projection onto a plane intersects. When they don't
exactly intersect at a point they can be connected by a line segment, the shortest line segment is unique. If this line segment is significantly short it can be considered as their intersection in 3D to compensate for numerical errors. The same idea has been implemented as part of this thesis work.

Fig 5.3 showing two points $P_a$ and $P_b$ on lines $P_1P_2$ and $P_3P_4$ respectively

The explanation given below will show how to compute this shortest line segment that joins two lines in 3D. In what follows a line is defined by two points $P_1$ and $P_2$, and any point $P_a$ on the line is given by the equation,
\[ P_a = P_1 + a (P_2 - P_1) \] \quad \longrightarrow (1)

Similarly a point on a second line \( P_b \) defined by two points \( P_3 \) and \( P_4 \) can be defined as,

\[ P_b = P_3 + b (P_4 - P_3) \] \quad \longrightarrow (2)

The values of the variables ‘a’ and ‘b’ range from negative to positive infinity. The line segments between \( P_1 \) \( P_2 \) and \( P_3 \) \( P_4 \) corresponds to the interval between 0 and 1 for the two variables respectively.

There are two approaches to find the shortest line segment between lines \( P_1 \)\( P_2 \) and \( P_3 \)\( P_4 \). The first is to write down the length of the line segment joining the two lines and then find the minimum. That is, minimize the following

Using equation (1) and (2) we have,

\[ \| P_b - P_a \|^2 \]

Substituting the equations (1) and (2) of the lines gives

\[ \| P_1 - P_3 + a (P_2 - P_1) - b (P_4 - P_3) \|^2 \] \quad \longrightarrow (3)

The above can then be expanded out in the \((x, y, z)\) components. There are conditions to be met at the minimum, the derivative with respect to ‘a’ and ‘b’ must be zero. These two equations can then be solved for ‘a’ and ‘b’, the actual intersection points found by substituting the values of constants into the original equations of the line.
An alternative approach but one that gives the exact same equations is to realize that the shortest line segment between the two lines will be perpendicular to the two lines. This allows us to write two equations for the dot product as

\[(P_a - P_b) \cdot (P_2 - P_1) = 0 \quad \rightarrow (4)\]

\[(P_a - P_b) \cdot (P_4 - P_3) = 0 \quad \rightarrow (5)\]

Expanding these given the equation of the lines,

\[(P_1 - P_3 + a(P_2 - P_1) - b(P_4 - P_3)) \cdot (P_2 - P_1) = 0 \quad \rightarrow (6)\]

\[(P_1 - P_3 + a(P_2 - P_1) - b(P_4 - P_3)) \cdot (P_4 - P_3) = 0 \quad \rightarrow (7)\]

Expanding these in terms of the coordinates \((x, y, z)\) is a very complex but the result as given by Bourke [38] is as follows,

\[d_{1321} + a d_{2121} - b d_{4321} = 0 \quad \rightarrow (8)\]

\[d_{1343} + a d_{4321} - b d_{4343} = 0 \quad \rightarrow (9)\]

Where,

\[d_{mnop} = (x_m - x_n)(x_o - x_p) + (y_m - y_n)(y_o - y_p) + (z_m - z_n)(z_o - z_p)\]

Note that \(d_{mnop} = d_{opmn}\)
Finally, solving for \( a \) gives

\[
a = \frac{d_{1343}d_{4321} - d_{1321}d_{4343}}{d_{2121}d_{4343} - d_{4321}d_{4321}}
\]

and back substituting gives \( \mu_b \)

\[
b = \frac{d_{1343} + ad_{4321}}{d_{4343}}
\]

Once we know the values of the constant ‘a’ and ‘b’ by replacing them in the line equation (1) or (2) will give us 2 points \( P_a \) and \( P_b \). By taking the midpoint of these points will give us the closest intersection point to these 2 lines.

That is,

\[
I = \frac{(P_a + P_b)}{2}
\]

where \( I \) is the intersection point.

5.3. Setup and results

The specimen used to validate the algorithm proposed in this work is a phantom dataset. This phantom dataset is composed of two different wires with different diameters which have been soldered together at some places to form an arterial-shaped configuration. It was created and scanned using a bi-planar x-ray system by professor Molloi from the University of California, Irvine. The reason behind using this dataset is that it looks similar to blood vessels and, most importantly, the geometric dimensions of this dataset are well known. Once the algorithm is validated for this phantom wire dataset, it can be applied to different,
more realistic medical datasets. This type of dataset is commonly used in the computer vision community for validation purpose. The dimension of the 2D phantom wire image is $512 \times 480$. Both left and right images are of the same dimensions.

Since the images are calibrated, we know specific parameter about the camera and the image locations. For example, the camera positions with respect to the each other and image planes. Isocenter is the location where the image center is located and it does not change even if the x-ray unit is moved. In this case, there are slight differences in the isocenter of left and right images. The isocenter of left and right images are at $(260, 236)$, $(255, 257)$ respectively.

After retrieving all this information, all we need to do is just set up the same environment composed of virtual cameras and image planes. The setup is as shown in the figure given below.
Fig 5.4 Environment set up showing the virtual cameras and image plane. In addition, it shows the specimen in the center.
As can be seen in the figure above, the specimen is in the center and the cameras are at a right angle to each other. The image planes are separated by a distance of $width \times 0.4230 \text{ (mm\_per\_pixel)}$. The cameras are at an equal distance from the image planes. Specifically in this case, they are 700 mms (millimeters) away from their image planes and the scaling factor is 0.4230 (mm\_per\_pixel).

Once everything is set up, we first scan through the pixels of left image checking for some specified threshold value and when a pixel value exceeds the threshold value we shoot a ray from the left camera position (x-ray source) to that particular pixel. After that we scan through the right image shooting rays from the right camera to the pixels exceeding the threshold value and check whether they intersect. These intersection points are indeed our volumetric data. These intersection points are mapped using memory mapping function during the extraction process described in the next section. Pictorial representation can be seen in figure 5.5.
5.3.1 Centerline Algorithm

To extract the exact 3D structure of an object, we need to know the centerline using which the diameter can be retrieved. The centerline algorithm used for this project is developed by Dr. Thomas Wischgoll. It is based on a topological analysis of a vector field derived from the normal vectors at the object’s boundary. Hence, in addition to the computation of the 3D volumetric points, normal vectors need to be calculated as described in the next section. The
following figure shows the centerline determined for a branch in phantom wire dataset,

Fig 5.6 centerline for a branch of phantom wire
5.3.2 Normal Vector

The normal vector, often simply called the "normal," to a surface or a point is a three dimensional vector perpendicular to it. For example, normal for a triangle can be calculated as,

\[
\text{edge1} = \text{point2} - \text{point1} \\
\text{edge2} = \text{point3} - \text{point1} \\
\text{normal} = \text{edge1} \times \text{edge2} \quad (\text{where } \times \text{ is the cross product})
\]

In our case, as it is a closed surface formed by point cloud, we need all the normals pointing inwards so that most of the opposite vectors cancel out each other. The method used to calculate normal vectors emulates the computation of gradient vectors in volumetric images. Direction vectors from a point to its neighboring points within a specified radius are computed and then averaged to get an estimate for the normal vector (explained later).

Finding the neighboring points for every individual point in the huge point cloud can be time consuming. To improve the performance of the implementation, an octree data structure is used. An octree is a tree data structure used to partition a three dimensional space by recursively subdividing it into eight octants. In an octree, each internal node has up to eight neighbors. When searching for neighboring points using the octree, the system will only consider points which are in the vicinity rather than searching through the entire point cloud.
Fig 5.7 showing few direction vectors for a point on the boundary.

The above figure shows how the direction vectors for some point in the vessel at the object’s boundary would look like. Averaging these direction vectors would give an estimate for the normal vector for that point pointing inside the vessel.
The algorithm for calculating normal vector for an individual is given below,

Algorithm

For every individual point

{ 
   
   For every neighboring point

   { 
      
      Direction_Vector = Direction_Vector + (Neighboring point – individual point)

      Add up all the Direction_Vector

   }

}

Length = Length (Direction_Vector);

Normal_Vector = Direction_Vector / Length;

Conclusively, using these normal vectors, the centerline algorithm is able to find out the centerline of the vessels.
5.3.3 Results and Snapshots

Fig 5.8 Left view of phantom wire image
The above figure (fig 5.8 and fig 5.9) shows the original phantom wire images taken by Professor Molloi from the University of California, Irvine. In our project, we refer to them as left and right images respectively. As we know the geometric dimensions such as diameter, length etc, of this dataset, we use it to validate our algorithm.
Fig 5.10 the above figure shows one view (left) of the phantom wire data along with the centerlines after rendered using 3D point cloud.

Fig 5.11 the above figure shows another view (right) of the phantom wire data along with the centerlines after rendered using 3D point cloud.
Fig 5.12 Reconstructed geometry of Phantom wire data (Left view)

Fig 5.13 Reconstructed geometry of a phantom wire dataset
As it can be seen in fig 5.12 and 5.13, we were able to extract most of the information from the original phantom wire data. We were able to achieve 90% of accuracy in terms of the diameter of the phantom wire, for both types of wires (Phantom wire model is constructed using 2 different wires of different diameters). The following figure shows how the diameter of the vessel or wire varies along the length. This graph is plotted for a segment of wire.

![Diameter vs Length of Phantom wire model](image)

**Fig 5.14 graph showing how the diameter varies along the length**

Experiments were also conducted on a set of real noisy medical images. The results and snapshots are given below.
Fig 5.15 Left view of a real medical dataset used for this experiment
Fig 5.16 Right view of a real medical dataset used for this experiment

The above two images (fig 5.16 and fig 5.17) are real images of arterial vessels. These images are used to reconstruct the 3D model.
Fig 5.17 Left view of real image dataset after rendered using 3D cloud points

Fig 5.18 Right view of real image dataset after rendered using 3D cloud points
Fig 5.19 Reconstructed geometry of a real dataset

Fig 5.20 Reconstructed geometry of a real dataset
Fig 5.19 and fig 5.20 shows the reconstructed geometry of a real medical dataset shown in the fig5.17 and fig 5.18. They are different views of the same dataset.

As it can be seen in both the images (fig 5.19 and fig 5.20), we are able to extract 3D information for most part of it except for the region where it is noisy. The level of accuracy achieved using this pair of medical data is around 80% which is less than what we achieved using phantom wire data and the main reason for this is noise in the real images.

![Diameter vs Length for Real Medical Data](image)

Fig 5.21 graph showing how the diameter varies along the length for the real medical image data

The above figure shows how the diameter of reconstructed 3D geometry of real medical data varies along the length of it major trunk. It can be seen in the fig that the diameter decreases as we go along the trunk of the arterial branch.
6. CONCLUSIONS AND FUTURE WORKS

The following section in this chapter presents a reflection on the results of the work presented.

6.1 Conclusion

Memory mapping was successfully implemented to facilitate efficient out-of-core functionality on ordinary off-the-shelf desktop computers with limited memory. The limitations in terms of data set size that can be processed are given by the hard disk size and the availability of address space. The size of the data that can be mapped can go up to 4 terabytes on a 64-bit machine using this method.

In addition, an algorithm was successfully implemented and tested for the reconstruction of a 3D object using calibrated orthographic images. Using this algorithm, we were able to achieve up to 90% accuracy for a phantom wire model.

6.2 Future works

The current implementation of the memory mapping function can be further tested by mapping multiple large datasets at the same time. Many different filters can be implemented and tested on different datasets. Also, the algorithm proposed for the reconstruction of 3D image can be experimented for more number of real medical datasets.
REFERENCES


[6]. Michael Beynon, Chialin Chang, Umit Catalyurek, Tahsin Kurc, Alan Sussman, Henrique Andrade, Renato Ferreira, Joel Saltz, Processing Large-Scale Multidimensional Data in Parallel and Distributed Environments Parallel
Computing (Special issue on Parallel data-intensive algorithms and applications), 28(5):827--859, May 2002


[10] Shannon Hastings, Stephen Langella, Tony Pan, Scott Oster, Umit Catalyurek, Joel Saltz, Tahsin Kurc, Managing Data and Image Processing Workflows for Large Scale Image Analysis


[40] http://developers.sun.com/solaris/articles/read_mmap.html
T10/node3.html