Visibility Driven Focus+Context Visualization of Multimodal Volume Data

by

Srinivas R. Vaidya (MT2013152)

A thesis submitted

in partial fulfilment of the

requirements for the degree of

Master of Technology

in

Information Technology

ज्ञानमुत्तमम्

International Institute of Information Technology, Bangalore.

June 2015

Thesis Certificate

This is to certify that the thesis titled **Visibility Driven Focus+Context Visualization of Multimodal Volume Data** submitted to the International Institute of Information Technology, Bangalore, for the award of the degree of **Master of Technology** is a bona fide record of the research work done by **Srinivas R. Vaidya (MT2013152)** under my supervision. The contents of this thesis, in full or in parts, have not been submitted to any other Institute or University for the award of any degree or diploma.

Prof. T K Srikanth

IIIT-Bangalore,

The 15th of June, 2015.

Abstract

The aim of this thesis is to study mechanisms to improve focus+context visualization of multimodal volumetric data. Specifically, the use of visibility histograms is investigated. Visibility histograms provide intuitive cues to the user (or the visualization application) about the contribution of particular scalar values to the final image. These cues are view dependent, and can be used to redesign transfer functions such that the visibility of regions of interest is improved, while maintaining surrounding context. We also use per-ray visibility histograms, which provides additional tools to examine multimodal volume data. The technique uses a ray tracing approach, and is applied to both single modality as well as dual modality data. The solution has been implemented on the GPU, to enable interactive response. The technique is demonstrated on multimodal data - CT and PET - to produce fused focus+context volume visualization, with focus on PET data, while the CT dataset provides clear contextual information.

Acknowledgements

" I would like to express my appreciation to all those who helped me during the course of this thesis work. My deepest thanks to my advisor, Prof. T K Srikanth for his continuous encouragement and valuable suggestions. It was my pleasure to work under his guidance. I also thank Prof. Jaya Sreevalsan-Nair for her insightful comments and suggestions."

"I would also like to thank my fellow GVCL labmates Amit Tomar and Shivam Agarwal for their moral support and remarkable encouragement. I also thank my parents for their unceasing encouragement, support and attention. I dedicate my work to my family and friends who have always been there to support and cheer me up throughout my ups and downs. "

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Visualization has become an indispensable tool in many areas of science and engineering. In particular, the advances in this field made over the past twenty years have turned visualization from a presentation tool to a discovery tool.

Volume visualization is a technique that enables physicians and scientists to gain insight into complex volumetric structures. Currently, the trend towards information acquisition using data sets from multiple modalities is increasing in order to facilitate better medical diagnosis. Volume visualization of such data helps radiologists to identify and quantify tumours from MRI and CT scans, and neuroscientists detect regional metabolic brain activity from PET and functional MRI scans. Analysis of these diverse types of images requires interactive visualization tools. The main focus of thesis will be on medical visualization. As different modalities frequently carry complementary information, our goal is to combine their strengths and generate visualizations that enable viewers to view the object of primary interest in greater detail, while at the same time not losing surrounding information or context, leading to better focus+context visualization and also improve usability by providing more intuitive image manipulation mechanisms.

1.1 Motivation

For volume models, the key advantage of using direct volume rendering is its potential to show the structure of the value distribution throughout the volume. The contribution of each volume sample to the final image is explicitly computed and included. The key challenge of direct volume rendering is to convey that value distribution clearly and accurately. In particular, we cannot achieve higher opacity and clarity for each volume sample, if volume samples in the rear of the volume are not to be completely obscured. Hence, many visualization techniques have been developed to enable viewing of volume data by adjusting (interactively or otherwise) the opacity and color of volume elements.

Despite the proliferation of volume rendering software, the design of effective transfer functions or the mapping of volume sample values to color and opacity in the image is still a challenge. The growing popularity of GPU-based volume renderers has advocated the use of a more exploratory approach, where use's can arrive at good transfer functions via trial-and-error modification of opacity and color values. However, effective transfer functions are often the product of time-consuming tweaking of opacity parameters until meeting a desired quality metric, often subjective. The reason to adopt such an is approach is partly due to lack of a metric to measure the quality of transfer functions.

With hardware acceleration, volume rendering has become very attractive for many applications. To be more widely adopted, however, its usability remains to be enhanced. In particular, the task of classifying volume data before rendering as well as the task of manipulating potentially a large number of rendering and viewing parameters to achieve desired visualization are often time-consuming and tedious.

In interpreting volume data for surgical planning or medical diagnosis, the information which can be visualized from a single modality, example, Computed Tomography (CT), may be insufficient. A number of factors influence this, including limited resolution, sensitivity to tissue properties, noise, etc. For this reason, radiologists often make use of additional modalities that provide complementary or supplementary information. In this way, radiologists are able to extract more clearly the structures of interest and the spatial relationships among them. For example, CT provides the most detailed anatomical information from the human body, usually at high resolution. It helps depict high dense structures such as bone, as well as the shape of internal organs. On the other hand, the acquisition of metabolic activity must rely on a modality like Positron Emission Tomography (PET). In general, metabolic activity is important to detect cancer, since cancer tumours and other malignancies are usually located in regions with high rate of metabolic activity, such as regions with high blood flow. To obtain the best of the two modalities, recent visualization systems attempt at fusing both types of information in a single meaningful image [14].

Multimodal volume rendering presents additional challenges, from the problems of superimposing dual modality data and highlighting objects of interest, to the desire to suppress occluding materials while maintaining the context and to enhance structural and spatial clarity of the objects.

The issue of visibility is not exclusive to medical data. Simulations of 3D phenomena often contain structures are intertwined in 3D space with other less interesting structures. Therefore, visualization of internal flow becomes difficult. The techniques discussed in this thesis can be applied to such datasets too.

1.2 Non Photorealistic Rendering

The emergence of non-photorealistic rendering (NPR) over the greater part of a decade has created an intriguing new field espousing expression, abstraction and stylisation in preference to the traditional computer graphics concerns for photorealism [8]. By lifting the burden of realism, NPR is capable of engaging with user's, providing compelling and unique experiences through devices such as abstraction and stylisation. Non-photorealistic rendering can be used to illustrate subtle spatial relationships that might not be visible with more realistic rendering techniques.

Volume rendering has remained a prevalent tool in medical and scientific visualisation for over more than a decade. The ability to visualise complex real-world phenomena has found its way into practical applications including CT and MRI scans of the brain and imaging flow in fluid dynamics. The integration of volume rendering with non-photorealistic rendering(NPR) is an intuitive and natural progression given the communicative and expressive capabilities of NPR [10].

Volume Non-photorealistic rendering acheives two complimentary goals: the communication of information using images, and rendering images in interesting and novel visual styles which are free of the traditional computer graphics constraint of producing "life-like" images [9]. Hence, Volume NPR techniques can be used to create visualizations of volume data that are more effective at conveying the structure within the volume.

1.3 Focus+Context for Volume Visualization

Visualization of CT, MRI or PET data allows physicians and radiologists to see internal structures and organs with much greater detail than with conventional methods. However, in some cases there is too much data to be displayed at once on a computer display (or the displays resolution may be insufficient for practical use). A simple and widely used solution is to apply a magnification factor to get closer to a specific region. But by doing so, it is equally easy to get lost in the dataset. This is generally called loss of context, because we are no longer able to visualize the entire dataset. When we zoom in, we are focusing on a certain feature that is of interest. Another approach to increase visibility of the ROI is to make occluding materials completely transparent. This brings focus to the region of interest, however, we could lose the context since the surrounding material or regions may become too transparent to provide meaningful information. In the field of Visualization this problem is called focus+context [20] and a number of successful solutions have emerged. The challenge is to find a way of looking at a high level of detail at this area of focus, without losing the overall context. We also want operations to be interactive and with better usability.

1.4 Related work

Many techniques for multimodal visualization render the multi-volume by mixing the component volumes at a certain step of the volume rendering pipeline, such as, accumulation, illumination or at pixel levels [3]. In another approach, one set of data and optical transformations are applied to the region of interest, and a different set of transformations to the remaining data. Few other techniques include interactive cuts [13], distorted views [22], opacity peelings [26] or ghosted views [11]. Opacity peeling, automatically finds the layers that compose an image from a given point of view. Cutaway views completely remove occluding, unimportant structures and possibly also remove valuable context information.

The technique of importance-aware rendering [21], can help visual hidden structure. Features within the volumetric data are first classified according to a new dimension, denoted as object importance, and the final image is generated by raycasting and combining the intersected features proportional to their importance (importance compositing). The object importance is added as a new dimension to the traditional volume rendering pipeline in order to maximize the visual information. This technique removes or suppresses less important parts of a scene to reveal more important underlying information.

Another technique similar to [21] is importance aware compositing [7], which uses a front-to-back sample composition equation for direct volume rendering that takes into account a measure of sample importance. Importance-driven techniques [21] and ghosted views [11] assign different opacities in a viewpoint dependent manner, so that the user constantly gets an uninterrupted view of internal structures. But both these technique require prior definitions of context regions.

1.5 Objective

The objective of this thesis is to study techniques for viewing of volumetric data with the ability to support interactive and intuitive mechanisms for adjusting opacities of volume elements, such that it enhances visibility of structures or regions of interest to achieve "focus+context". Quantification of visibility of structures or regions relative to the entire model is studied, which provides intuitive insights while designing transfer functions. Specifically, the technique described in [5] and [25] is analyzed. The algorithm used to implement is parallelizable and is implemented on the GPU, which leads to enhanced performance, better interactivity, and good user experience.

1.6 Organization of thesis

The thesis is divided into two parts. The first part deals with visibility histograms, which represents the visibility of the sample values from a given viewport. The visibility histogram provides a feedback mechanism for designing transfer functions. Visibility histograms are view and opacity dependent. This method becomes an important aid for volume exploration. Therefore, in the first part of thesis we deal with generation of visibility driven transfer functions.

The second part of deals with challenges posed by multimodal visualization to generate informative pictures from complementary data (we use CT and PET datasets). The visibility information is used for generating focus+context visualization of fused multimodal datasets. Using visibility calculations, tradeoff between visibility and spatial clarity is handled.

Chapter 2

Direct Volume rendering using raycasting

A 3D dataset normally consists of around one hundred slices, but in some cases, it can be many hundreds of slices. In medical usage, it is a normal practice for radiologists, or medical practitioners to visually inspect each slice [17]. The common landmarks, such as the major blood vessels or skeleton will be identified, and the locations of the abnormalities will be determined based on these landmarks. Then, they need to mentally visualize the anatomy of the patient with the associated abnormalities, a task that needs a lot of experience and knowledge. Thus, the process of identifying the structures based on the 2D slices is very tedious, time consuming, and prone to error and could impact the decision on treatment planning. Therefore, volume rendering techniques are needed in order to fully utilize the information of 3D datasets.

The term volume rendering is used to describe techniques which allow the visualization of three-dimensional data. It is a technique for visualizing sampled functions of three spatial dimensions by computing 2-D projections of a colored semi-transparent volume, as shown in figure 2.6. In scientific visualization and computer graphics, volume rendering is a set of techniques used to display a 2D projection of a 3D discretely sampled data set, typically a 3D scalar field. 3D datasets can be structured, unstructured or hybrid grids.

2.1 Introduction

Classification is a term that refers to assignment of optical properties to data values. Classification is one the most important steps in the volume rendering pipeline, since it is these optical properties that will either emphasize an feature or de-emphasize it. The assignment of optical properties to data values is accomplished using a transfer function. Some applications of classification are:

- Isosurfaces can be shown by mapping boundary elements of corresponding data values to almost opaque values and the rest to transparent values. The appearance of surfaces can be improved by using shading techniques to form the RGB mapping.

- Opacity can be used to see the interior of the data volume too. These interiors appear as clouds with varying density and color. A big advantage of volume rendering is that this interior information is not thrown away, so that it enables one to look at the 3D data set as a whole.

2.2 Definitions

2.2.1 3D Grid

3D datasets can be structured, unstructured or hybrid grids.

2.2.1.1 Structured Grids

Structured grids are identified by regular connectivity. The possible element choices are quadrilateral in 2D and hexahedra in 3D. This model is highly space efficient, i.e. since the neighborhood relationships are defined by storage arrangement.

2.2.1.2 Unstructured Grids

An unstructured grid is identified by irregular connectivity. It cannot easily be expressed as a two-dimensional or three-dimensional array in computer memory. This allows for any possible element that a solver might be able to use. Compared to structured meshes, this model can be highly space inefficient since it calls for explicit storage of neighborhood relationships.

2.2.1.3 Hybrid Grids

A hybrid grid contains a mixture of structured portions and unstructured portions. It integrates the structured meshes and the unstructured meshes in an efficient manner. Those parts of the geometry that are regular can have structured grids and those that are complex can have unstructured grids. These grids can be non-conformal which means that grid lines dont need to match at block boundaries.

This thesis focuses on structured grids with hexahedral elements which are of uniform size.

2.2.2 Voxel

A voxel represents a single sample, or data point, on a regularly spaced, three-dimensional grid. It is the basic element of the volume. This data point can consist of a single piece of data, such as intensity or opacity, or multiple pieces of data, such as a color in addition to opacity. A voxel represents only a single point on this grid, not a volume; the space between each voxel is not represented in a voxel-based dataset. The value of a voxel may represent various properties. In CT scans, the values are Hounsfield units, giving the opacity of material to X-rays [16].

2.2.3 Direct Volume Rendering

Direct volume rendering involves generating visualizations without creating intermediate geometric structures, such as polygons of isosurfaces, but simply by a direct mapping from volume data points to composited image elements [18]. Together with traditional computer graphics elements such as camera, lighting, and shading, the central ingredient in this direct mapping is the assignment of optical properties (opacity, color, etc.) to the values comprising the volume dataset [1].

2.2.4 Transfer Functions

The role of the transfer function is to emphasize features in the data by mapping values and other data measures to optical properties. The simplest and most widely used transfer functions are one dimensional, and they map the range of data values to color and opacity, as shown in figure 2.1. For correct rendering, the color components need to be multiplied by the opacity, because the color approximates both the emission and the absorption within a ray segment, and this is refered to as opacity-weighted color [24].

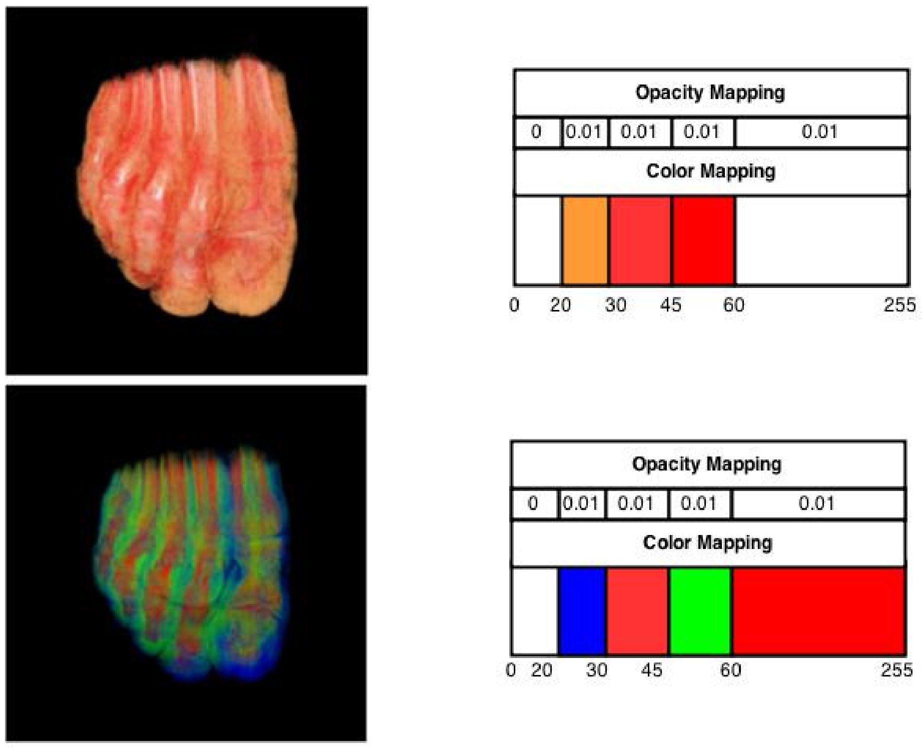


Figure 2.1: Application of two different 1D transfer functions on the same dataset. The transfer functions are shown next to their corresponding image.

2.2.5 Region of Interest

A region of interest (often abbreviated ROI), is a selected subset of samples within a dataset identified for a particular purpose. A ROI can be any of the following.

- Volume of Interest, which is specified as a range of voxels within an 3D volume. For example, we could define a cubic, spherical or cylindrical ROI.
- A range of voxel intensities can also be a ROI.
- A range of voxel intensities of one modality can be defined as an ROI in a multi-modal setup.

2.3 Volume Ray Casting

Raycasting is a technique to visualize volume data. It is method in which for every pixel in the image, a ray is cast through the volume parallel to the view direction. The ray intersects(or passes close to) a line of voxels. The color of the pixel is computed based on color and transparency of the voxels that are intersected by the ray.

The ray equation is defined by a starting point and a direction, as shown in figure 2.2. If the ray hits the volume, the color of the pixel is calculated by sampling the data values of the ray at a finite number of positions in the volume. On each sample the transfer function is applied and composited with accumulated values of the ray.

2.3.1 Basic Algorithm

In its basic form, the volume ray casting algorithm comprises of a ray being cast through the volume, for each pixel of the final image. Generally, the volume is enclosed within a bounding primitive, a simple geometric object – usually a cuboid – that is used to intersect the ray of sight and the volume. Along the part of the ray of sight that lies within the volume, equidistant sampling points or samples are selected.

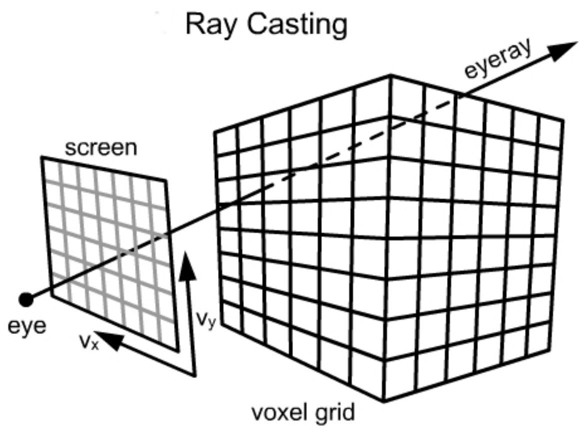


Figure 2.2: Direct volume rendering using raycasting.

In general, the volume is not aligned with the ray of sight, and sampling points will usually be located in between voxels. Because of that, it is necessary to interpolate the values of the samples from its surrounding voxels. After all sampling points have been fetched, they are composited along the ray of sight, resulting in the final color value for the pixel that is currently being processed. This compositing technique is explained in detail in a later section.

2.3.2 Two-pass Rendering

To perform raycasting on a volume for an arbitrary view direction, we need to compute the points at which each ray enters and exits the cube corresponding to the volume. An efficient and convenient method for this is to render the front and back faces of the cube into separate images, as shown in figure 2.4. The same pixel coordinates on the two images correspond to the images of points on a given ray, and specifically the entry and exit points of the ray. Thus, by tracking the mapping from

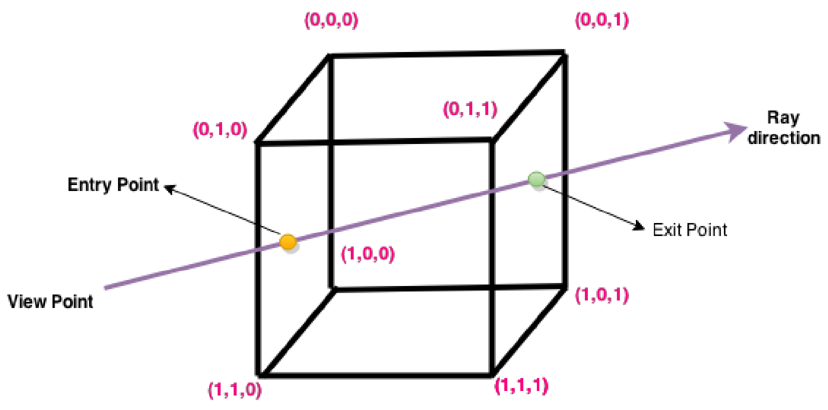


Figure 2.3: Assigning unique colors to all corners to the bounding box.

points on the cube faces to the pixels on the images, we can compute the entry and exit point for each ray.

Pseudocode of Two-pass Volume ray casting:

```
First pass: render the backface of the boundbox
Second pass: render the frontface of the boundbox
Lookup volume exit position from backface 2D texture
Entry position obtained at second pass
Compute ray of sight direction

While in volume
        Lookup data value at ray position
        Apply transfer function to data value
        Accumulate color and opacity
        Advance along ray
```

We use two float-textures, where the color value encodes the co-ordinates of the entry and respectively the exit points. To obtain intersection of the ray of sight with data volume, we use a bounding box, which is a cube with each edge of unit length. Each of the eight corner vertex of the cube are assigned unique colors, as shown in figure 2.3. During rendering, OpenGL will interpolate the color values between the vertices automatically. Since the corner colors are unique, we can use the interpolated colour values of a pixel to reconstruct the 3D coordinates of the intersection points of the ray of sight. On finding the entry and exit points on the cube, ray direction is easily obtained. From the entry point, ray starts marching towards the exit point while sampling points at equidistant locations.

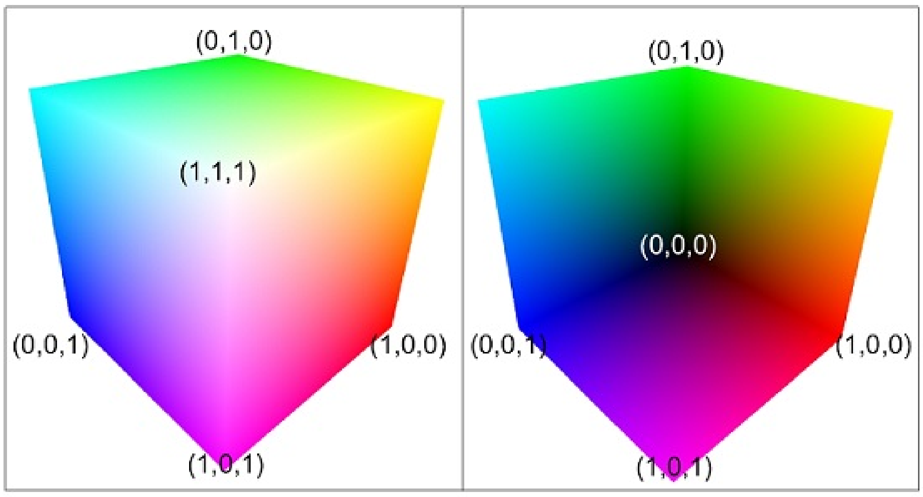


Figure 2.4: Entry and exit Textures.

2.3.3 Sampling

Along the part of the ray of sight that lies within the volume, equidistant sampling points or samples are selected. In general, the volume is not aligned with the ray of sight, and sampling points will usually be located in between voxels, as shown in figure 2.5. Because of that, it is necessary to interpolate the values of the samples from its surrounding voxels.

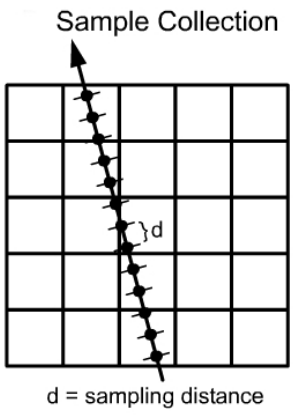


Figure 2.5: Sampling along the direction of ray.

2.3.4 Compositing

The discrete version of the volume rendering equation replaces the continuous integral with a Riemann sum.

Discrete Volume Rendering Equations:

$$C = \sum_{i-1}^{n} C_i \prod_{j=1}^{i-1} (1 - A_j) \tag{2.1}$$

$$A = 1 - \prod_{j=1}^{n} (1 - A_j) \tag{2.2}$$

Here, Opacity A_i approximates the absorption and opacity-weighted color C_i approximates the emission and the absorption along the ray segment between samples i and i+1.

This formula is efficiently evaluated by sorting the samples along the viewing ray and computing the accumulated color C and opacity A iteratively. The color of each sample is determined by classification and shading, and when the transparency is determined by classification, the next step is to evaluate the volume rendering equation using these values. This evaluation is performed by a process called compositing which generates the final pixel value for a ray shot into the volume. Two techniques for compositing are front-to-back and back-to-front compositing.

2.3.4.1 Back-to-Front Compositing Equations

In this technique, samples are sorted in back-to-front order, and the accumulated color and opacity are computed iteratively. A single step of the compositing process is known as the Over Operator.

$$\hat{C}_i = C_i + (1 - A_i)\,\hat{C}_{i+1} \tag{2.3}$$

$$\hat{A}_i = A_i + (1 - A_i)\,\hat{A}_{i+1} \tag{2.4}$$

where, C_i and A_i are the color and opacity obtained from the fragment shading stage for the ray i, along the viewing direction, and $\hat{C}_i$ is the accumulated color from the back of the volume.

2.3.4.2 Front-to-Back Compositiong Equations

In this technique, samples are sorted in front-to-back order. Front-to-back compositing has an added advantage over back-to-front compositing. Since the composition is done towards the back and the current transparency is known at all times, the compositing can be stopped early when the opacity is above a threshold (since nothing behind that point is going to affect the final image). This is one of the more powerful optimizations that can be done for several rendering techniques.

The front to back equation is:

$$\hat{C}_i = (1 - \hat{A}_{i-1})\, C_i + \hat{C}_{i-1} \tag{2.5}$$

$$\hat{A}_i = (1 - \hat{A}_{i-1})\, A_i + \hat{A}_{i-1} \tag{2.6}$$

where $\hat{C}_i$ *and* $\hat{A}_i$ are the accumulated color and opacity from the front of the volume.

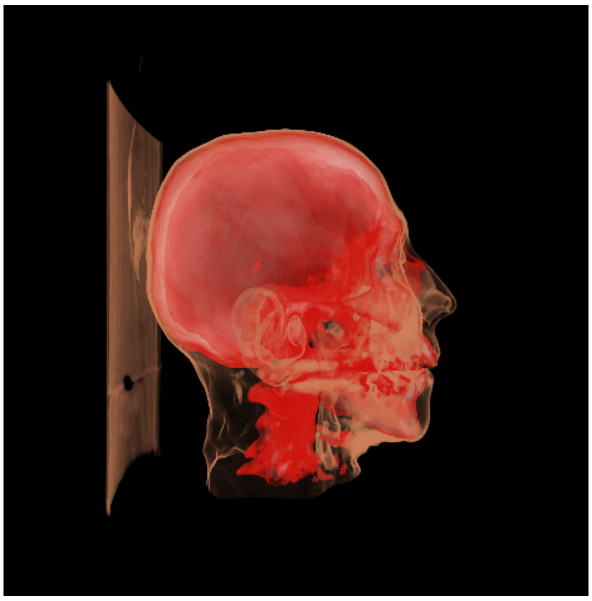


Figure 2.6: Direct volume rendering of a volumetric head dataset.

Chapter 3

Visibility Driven Transfer functions

3.1 Motivation

Medical imaging has given radiologists an ability that photography was not able to provide, it lets them see inside the human body. With the advent of 3D visualization systems, these images can be put together into crisp and impressive renderings of the human body from a variety of perspectives that were only dreamt of before, revolutionizing clinical practice.

Light transport models soon emerged to allow light interactions that, although not realistic in the physical sense, proved to be more effective for understanding the complex relationships among the anatomical structures. For instance, bone could be made semi-transparent to provide visibility of brain tissue. Skin could be removed altogether from an image to show only muscle or internal organs. However, soon it became evident that simply rendering these images in their raw form was no longer effective and the clear visualization of internal structures remains elusive.

The depiction of internal parts in the context of the enclosing space is a difficult problem that has occupied the mind of artists, illustrators and visualization practitioners. Despite the advances made in computer graphics for simulating the light transport in semi-transparent media, the problem of visualizing internal objects is no longer a rendering problem, but that of classification. Medical imaging technology obtains representations of anatomical structures via indirect ways, such as the response of tissue to X-rays or the alignment of electrons in a magnetic field. Therefore, the absence of semantic information prevents visualization practitioners from clearly marking up the regions that must be visualized. Without access to those regions, exploration becomes tedious and time-consuming. The predominant approach has been the use of transfer functions, or opacity mappings, which assign transparency and color properties to different intervals in the data. This method, however, does not guarantee visibility of internal structures or structures of interest in the volume. There is need to incorporate a measure for visibility, which gives contribution of each volume element in making of final image. In this chapter, visualization techniques to obtain clear views of internal features in 3D volume data are discussed along with visibility metric.

3.2 Related Work

With the fast growth in computational power of graphics hardware, it has recently become possible to manipulate 3D volume data in fashions that were only possible for surface meshes and CAD models, where semantic information is often explicit and readily available. When volume data are understood as explorable objects, it can be disassembled into parts that can be decomposed in numerous ways. One of the foremost ideas that were explored in this direction were cutaways, where certain parts can be removed to uncover hidden parts of the 3D volume [23]. Exploded views extend this idea to reveal the relationships among the internal parts of a complex volume [2], as shown in figure 3.1.

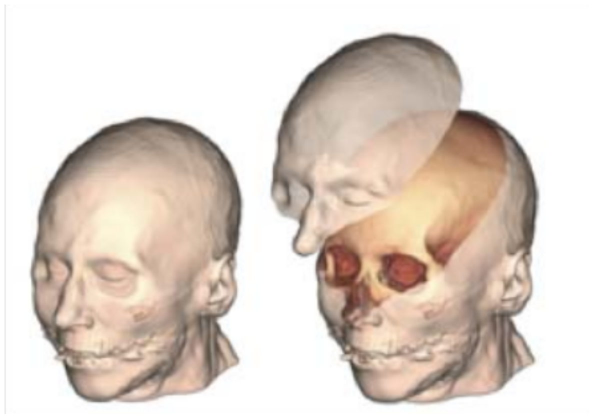


Figure 3.1: Cutaway of the skin of a CT data set [2].

Another strategy is to assign material properties to different regions or layers of a volume and simulate the physical response to the deformation and cutting of such regions [15]. Figure 3.2, for example, shows the result of simulating a peel away on a CT scan of a piggy bank to reveal a number of coins in its interior.

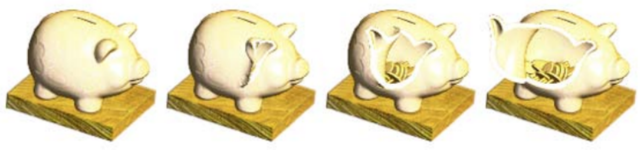


Figure 3.2: Volume peeling of a CT scan of a piggy bank [6].

Rigid and deformable cuts, although effective for visualizing the internal structures, work under the premise that the internal and external layers are clearly separated. In a more general sense, this separation is not easy to come by, and, in most cases, there is a degree of uncertainty. For this reason, the effective visualization of internal structure must rely on robust classification.

The main challenge when attempting to see the internal features remains that of classification. An effective visualization must first decide what regions it must preserve, and what regions are unimportant. However, volume data seldom contain any semantics about the captured structures. Acquisition technology outputs a series of images with intensity values, while simulations of 3D phenomena sample a continuous scalar or vector field in a grid. Therefore, traditional classification systems, found in off-the-shelf visualization systems, only consider a single dimension for classification, without regards of the spatial characteristics or the semantics of the data.

In an attempt to extract semantic information, one may analyze the spatial properties of the data, such as the location of boundaries [12], regions of high curvature [4], or shape [19]. In most of these cases, these properties are just approximations of the local distribution of data in a small neighborhood. Size, for example, can be measured as the extents of regions of certain homogeneity. Regions of a certain material, such as brain, that occupy a large volume, have different properties than those regions, such as skull and skin, that are relatively thin.

These observations have enabled us to construct classification based on size, and assign opacity based on the relative thickness of features. A particular example is the visualization of brain MRI, where the data is comprised of a series of thin layers (i.e, skin, skull and tissue) surrounding a large region, the brain, of a certain material.

Exploiting these properties lets us minimize the effects of occluding tissue, such as skin, to reveal the brain tissue clearly, as shown in figure 3.3.

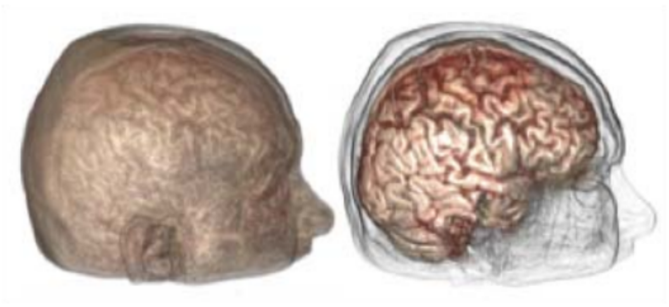


Figure 3.3: Size-based transfer function to visualize the brain. On the left is the original rendering, where brain is obscured by the skin [4].

3.3 Visibility histogram

3.3.1 Visibility

One of the limitations of contemporary visualization systems is the inability to quantify how visible a feature of interest is. To be more effective, along with traditional transfer function design, visualization systems must also incorporate a measure of visibility.

Visibility Metric, as defined in [5], attemps to measure the impact of individual samples on the image generated by a volumetric object. It is measured as the contribution of a structure of interest to the final image. Here, visibility can be used to quantify the quality of transfer function and ease their design towards more meaningful and efficient visualization. Transfer functions generated with this approach are called as visibility driven tranfser functions.

This process measures visibility of all structures in a volume to arrive at a good transfer function. In general, a visibility-driven transfer function is constructed in such a way that it attempts to guarantee the visibility of all structures of a volume and at the same time maximizes the visibility of structure of interest, in particular, of those features lying at the interior of a data set.

3.3.2 Visibility Histogram

The contribution of a sample in the volume to final image is referred to as visibility of that sample.

$$\alpha(s) \ = \ 1 \ - \ e^{\int_s^D \tau(t)dt} \tag{3.1}$$

where $\tau(t)$ is the attenuation coefficient of a sample, usually represented as an opacity transfer function $\mathcal{O}$ which is defined by user. Visibility also depends on viewpoint as accumulated opacity in front of the sample may differ at different viewpoints.

A visibility histogram is a representation of distribution of the visibility metric in relation to the domain values of the volume. Samples are weighted by visibility and added into bins that partition the range of values in the scalar field.

$$VH(x) = \mathcal{O}(x) \int_{s \epsilon \omega} \delta(s,x)(1 \ - \ \alpha(s))ds \tag{3.2}$$

where, $\delta(s,x)$ is a function.

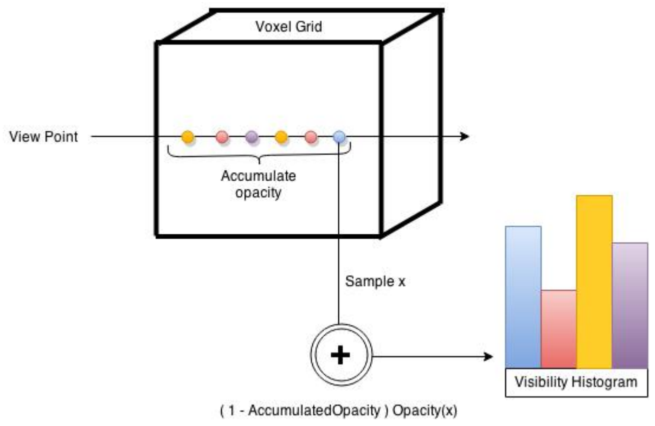


Figure 3.4: Visibility histogram computation using Raycasting.

$$\delta(s,x) = \begin{cases} 1 & V(s) = x \\ 0 & Otherwise \end{cases} \tag{3.3}$$

Hence, for all sample values x in the volume.

$$VH[x] = VH[x] + (1 - AccumulatedOpacity)Opacity(x) \tag{3.4}$$

3.3.3 Influence of the Transfer Function

Visibility histograms are transfer function dependent. A visibility histogram depicts the visibility distribution for a given data set only with respect to a certain transfer function. Every time the alpha transfer function changes, the visibility histogram has to be recomputed. Figure 3.5 shows how different transfer functions applied to the same dataset can lead to different visibility distributions.

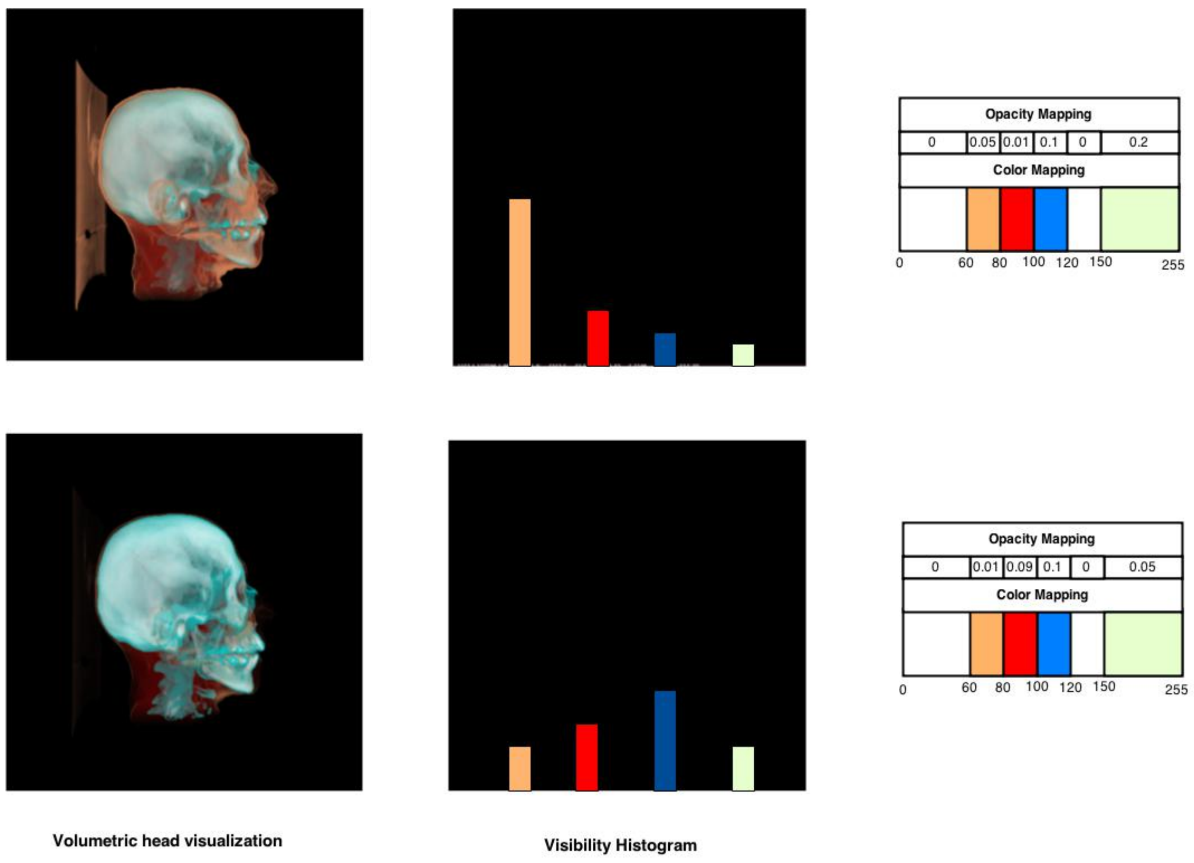


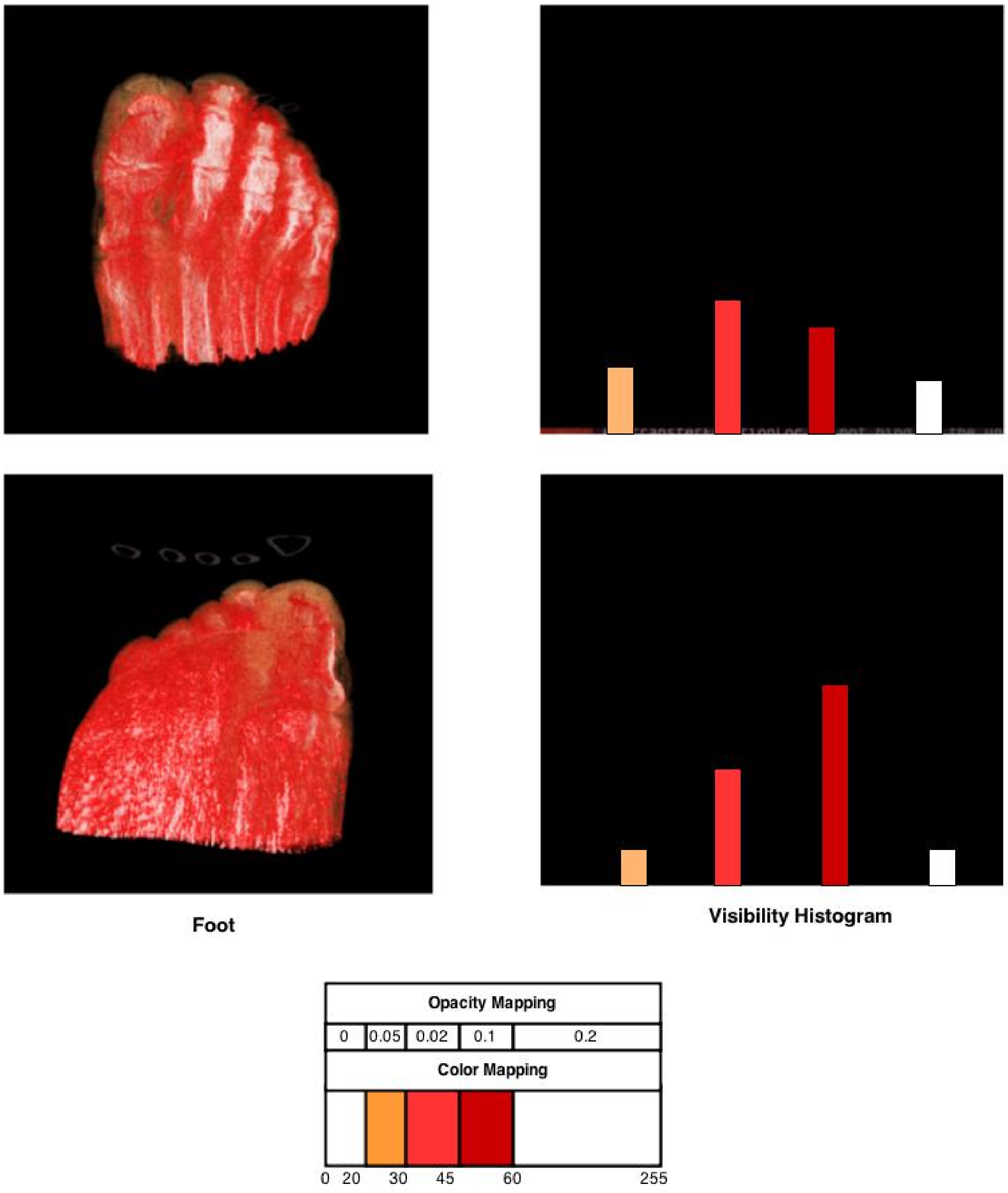
Figure 3.5: Illustration of influence of opacity mapping on the visibility histogram when viewing direction and sampling stepsize is constant.

3.3.4 Influence of the Viewing Direction

A visibility histogram for a particular dataset is dependent on the direction from which the dataset is looked at, i.e. the direction of the viewing rays which traverse the data set. On rotating the dataset, the sample points that are encountered along the viewing rays change, causing a change in the visibility distribution, as shown in figure 3.6.

3.4 Role Of Raycasting In Computing Visibility Histogram

As discussed in previous chapter, section 2.3, we know that basic idea behind volume ray casting is to shoot viewing rays through the data volume for every pixel in the image plane, and samples are taken at evenly spaced points (see figure 2.5) and blended together using the front-to-back compositing scheme defined by equation 2.5 and 2.6 to calculate the intensity value for the pixel in the image plane from which the ray originated.



Opacity Mapping				
0	0.05	0.02	0.1	0.2
Color Mapping				
⬜	🟧	🟥	🟥	⬜

Figure 3.6: Illustration of influence of viewing direction on visibility histogram when transfer function and sampling stepsize is constant.

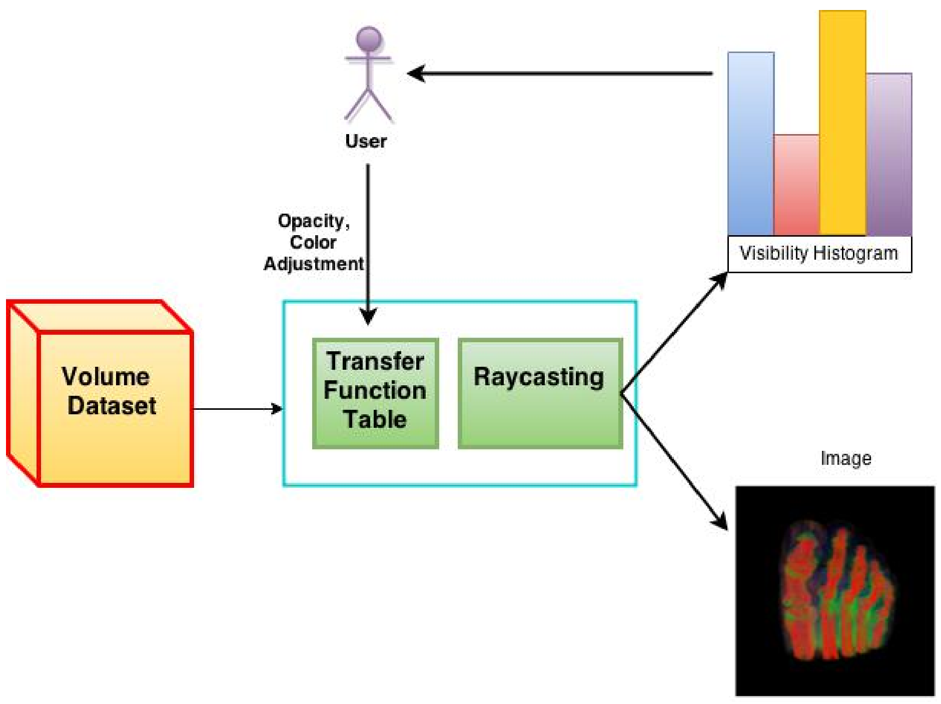


Figure 3.7: Transfer function design using visibility histogram.

3.5 Transfer Function Design

3.5.1 Manually Generated Transfer Functions

Trial and error is used commonly for finding an appropriate transfer function for a certain dataset. This approach is easy to handle, but can be very time consuming. The transfer function is typically generated by the user. Then, the dataset is rendered with the specified transfer function. If the resulting image does not meet the userś expectations, the transfer function is modified. This keeps untill user's expectation is met. The problem with this approach is that the user has lots of possibilities for generating and modifying transfer functions, and it might take a huge number of steps to find an appropriate one.

To address this issue, visibity histogram provides visual cues on contribution of samples in making of the final image. The visibility histogram itself is important as a visual aid.

As shown in the figure 3.7, the volumetic data is rendered using volume raycasting with the transfer function defined by user to generate final image and it's corresponding visibility histogram. The visibility histogram, which is generally plotted as a graph, provides immediate feedback to user regarding contributions of volume elements in the final image. The Visibility histogram helps find occulsion patterns on the data, as shown in figure 3.8, which helps in opacity modulation in a intuitive fashion to generate user desired results. Since, feedback mechanism is interactive, opacity modulation is easier and desired images are generated quickly.

3.5.2 Semi-automatic Transfer Functions Design

A further goal is to automate design of transfer functions that would result in improved visibility of the regions of interest. This has been formulated as an energy minimization problem [5]. The following energy components, designed to highlight certain desired aspects of the transfer function are:

- User-satisfaction: To ensure user-satisfaction, we minimize the mismatch between the computed opacity function and the original one defined by the user.
- Visibility: The second component is used to maximize the visibility of a given sample.
- Constraint: These constraints are used to control the minimum and maximum values opacities of value intervals in the final opacity function.

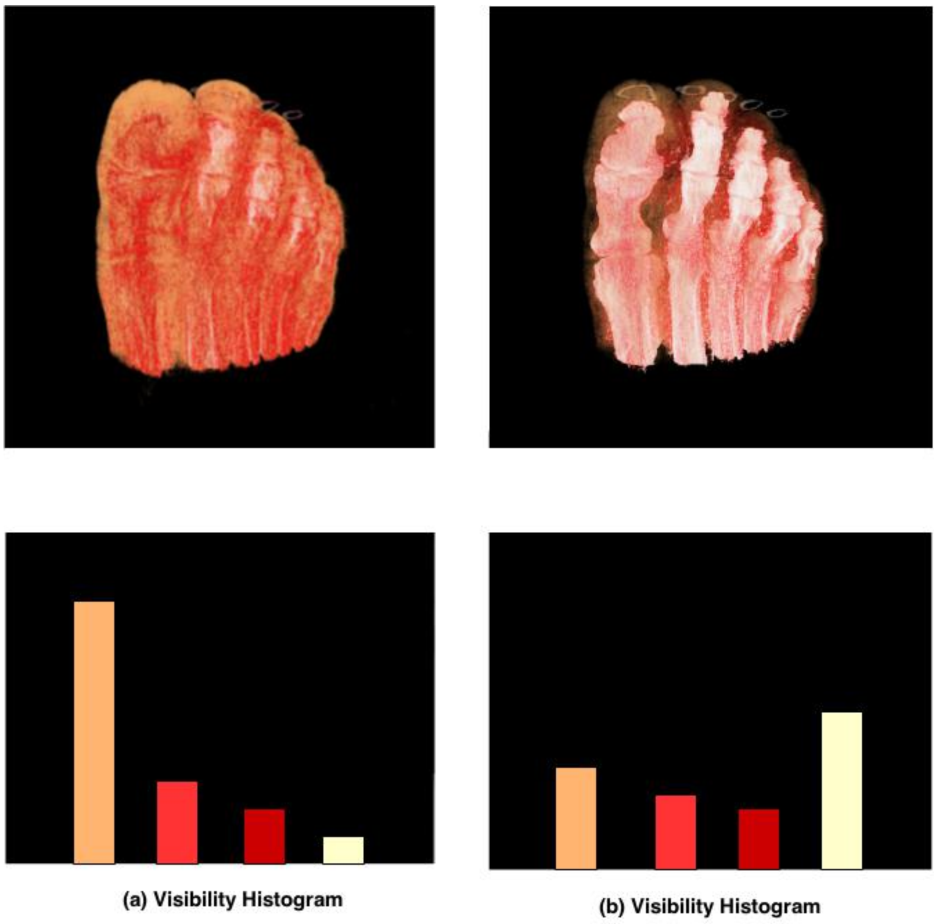


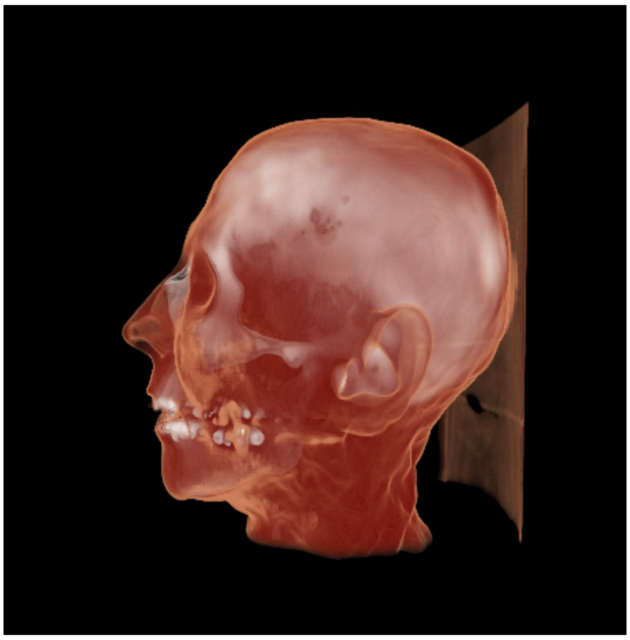
Figure 3.8: Visibility histogram depicting occlusion signature of a foot dataset. (a) The signature on the histogram reveals presence of a strong occluder. (b) The signature of this histogram reveals no presence is a strong occluder.

In the thesis, we have not focused on semi-automatic transfer function design aspect.

3.6 Focus+Context Using Visibility Histograms

A region of interest can be a range of voxel intensities, and this segmentation is achieved using transfer function. From section 3.3.2, we know that visibility histogram gives graphical cues on visibility of samples in making of the final image. Since range of voxel intensities define the ROI, visibility histogram can be used to find visibility of the ROI. Visibility histogram also reveals the occluding patterns, as illustrated in figure 3.8. Using the visual cues from the Visibility Histogram, the user can modulate opacities to set the focus on ROI. Tradeoff between visibility and spatial clarity can be handled, by adjusting opacities of regions surrounding the ROI untill it meets user requirement.

A region of interest can also be a volume, which is set of voxels within an 3D volume. Figure 3.9, illustrates volume ROI on volumetric head dataset, figure 3.9(a) shows original rendering and figure 3.9(b) shows volume ROI defined in the neck region, allowing the visualization of insides of the neck area.


(a) Original

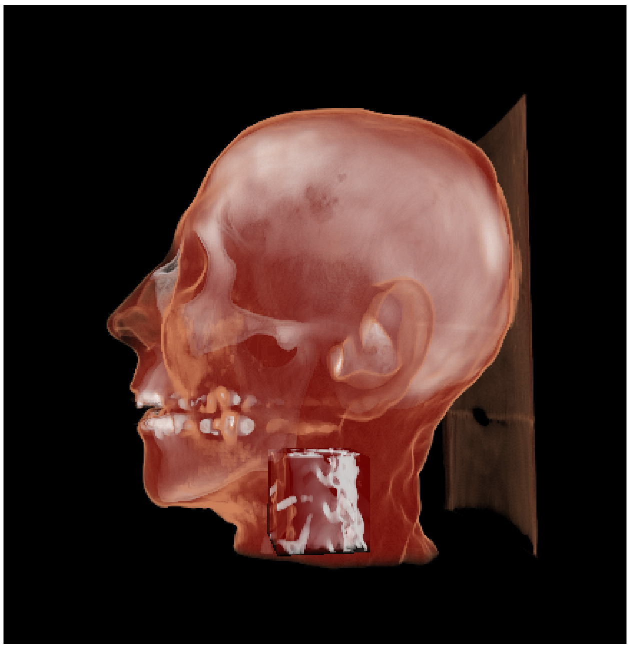

(b) Volume-defined ROI

Figure 3.9: Illustration of Volume-defined ROI

Chapter 4

Visibility guided multimodal volume visualization

4.1 Motivation

It is often desirable to visualize multimodal data by rendering the multiple volumes into one image. The general approach to this visualization is to fuse the multiple datasets based on their weighted intensity values. For example, CT is usually used to give us contextual information, to show anatomical relationship between pathological tissue (highlighted with PET data) and other non-effective regions. However with this approach, the intensity of the target area is often similar to that of neighbourhood resulting in lack of focus and clarity of sections of interest. Another approach when intensities are similar, is to manually segment out a part, and define a separate transfer function for highlighting the target area. Such manual tasks are tedious and error-prone.

4.2 Context+Focus Visualization For Multimodal Volume

The advantage of using additional modalities is that they provide complementary or supplementary information. However, handling of multiple sources for a structure, and generating a single image from them, is quite challenging.

The notion of visibility, as discussed in the previous chapter, helps us to summarize distribution for visibility a structure of interest from a given viewpoint. For example PET and CT dataset, PET data acts as the ROI we need to focus and CT data gives us the context. The techniques discussed in the previous chapter can used to modulate CT data opacities to obtain "focus+context" visualization.

4.3 Local Visibility Histogram

One technical challenge that lies in multimodal dataset visualization is in ROI definition. One way would be set up a ROI around the structure of interest manually by identifying some key boundary points. If the structure is complex, user needs to define a larger number of boundary points around the ROI, which is time-consuming and error-prone. One solution to this problem is to define a sphere or bounding box covering the ROI, as shown in figure 4.1. This solution is difficult to adopt, when the structure is complex, scattered or noisy.

One approach to increase visibility of ROI is to make occluding materials completely transparent, but this completely loses all the context information. Hence, for those rays hitting the ROI, local histograms is computed for each ray, as shown

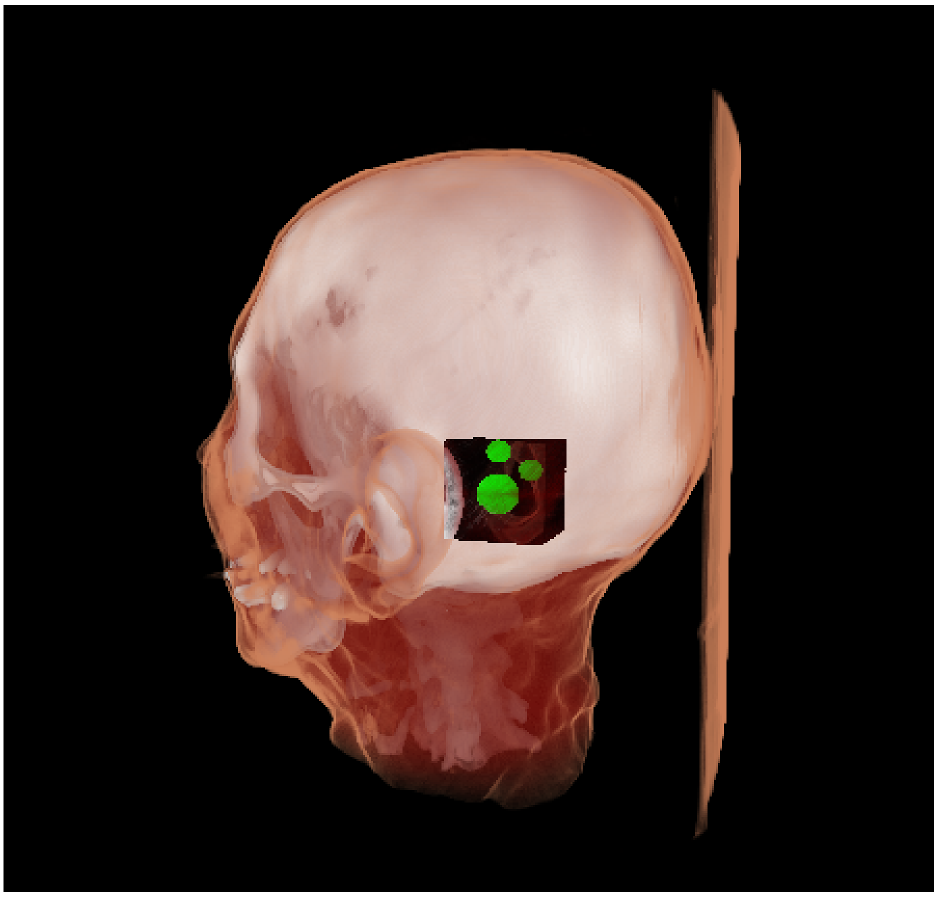


Figure 4.1: ROI bounded by a volume.

in figure 4.2. Rest of the rays cast into the volume are used to compute the global histogram. Using this, the user will now be able to manipulate the area of ROI, and control the visibility of this area. Upon varying the view direction local histogram is recalucated.

Once visibility histogram is known, opacity of samples in the ROI are adjusted using the formula 4.1.

$$A'(x) \;=\; A(x)\,(\,1\;-\;VH(X)\,)^{e} \tag{4.1}$$

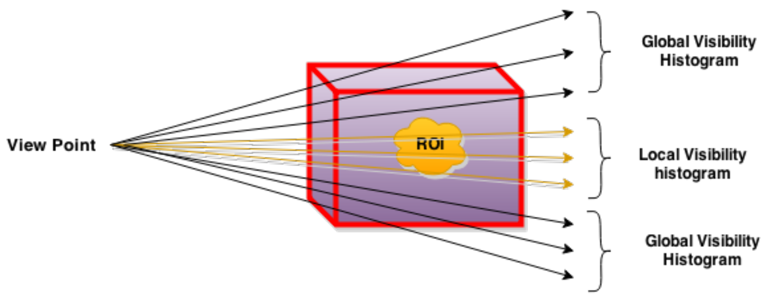

Global Visibility Histogram
View Point
ROI
Local Visibility histogram
Global Visibility Histogram

Figure 4.2: Global histogram and local histogram.

where, A(x) and A$'$(x) are the old and new opacity mappings for intensity x. VH(x) is the visibility associated with that intensity in the visibility histogram. e is the exponent that defines the strength of such a mapping, as illustrated in the figure 4.3.

From equation 4.1, when VH(x) is high(ie., it is more visible) it is likely to be made more transparent. When VH(x) is low, these samples do not occlude much and are retained to provide context. This helps in keeping context clear which increasing the focus.

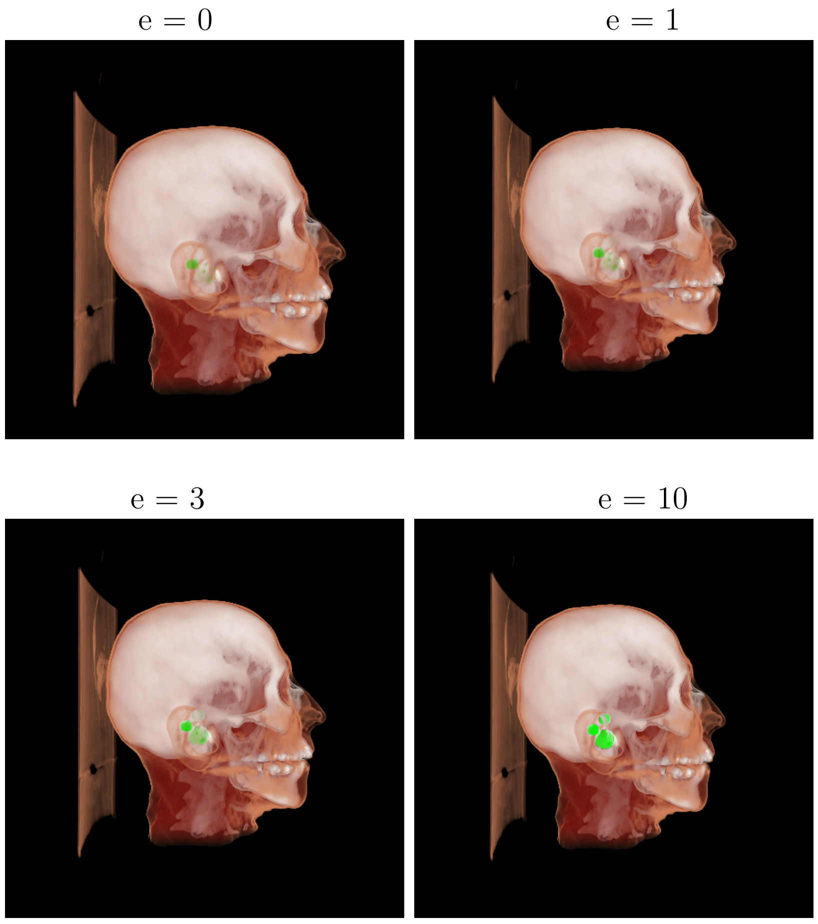


Figure 4.3: Effects of visibility-weighted adjustment e.

Chapter 5

Implementation

5.1 Introduction

5.1.1 The Graphics Pipeline

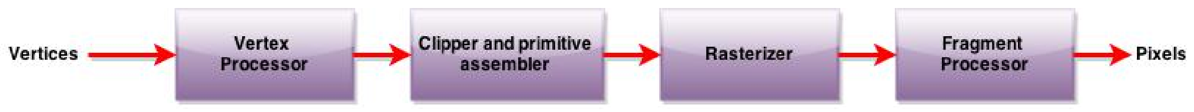


Figure 5.1: Graphics rendering pipeline overview.

In 3D computer graphics, the graphics pipeline or rendering pipeline refers to the sequence of steps used to create a 2D raster representation of a 3D scene. The figure 5.1, shows four major steps in the imaging process.

- **Vertex Processing:** In the first block of our pipeline, each vertex is processed independently. Here, a vertex is a set of attributes such as its location in space, as well as its color, normal, texture coordinates, amongst others. The inputs for this stage are the individual vertices attributes. The two major functions of this stage are to carry out coordinate transformations and to compute a color for each vertex. Many of the steps in the imaging process are transformations between representations of objects in different coordinate systems.

- **Clipping and primitive assembly:** The inputs for this stage are the transformed vertices, as well as connectivity information. The connectivity information tells the pipeline how the vertices connect to form a primitive. It is in here that primitives are assembled. This stage is also responsible for clipping operations against the view frustum, and back face culling. We perform clipping because of the limitation of the viewing window size.

- **Rasterization:** The primitives that emerge from the clipper are still represented in terms of their vertices and must be converted to pixels in the frame buffer. For example, if three vertices specify a triangle with a solid color, the rasterizer must determine which pixels in the frame buffer are inside the polygon.

- **Fragment processing:** The final stage in our pipeline takes in the fragments generated by the rasterizer and updates the pixels in the frame buffer. If the application generated three-dimensional data, some fragments may not be visible because the surfaces that they define are behind other surfaces.

Advancements in graphics cards have given programmers the ability to define the functionality of two of the above described stages: Vertex shaders for the Vertex processing stage, and fragment shaders to replace the fragment processing stage of fixed functionality.

These shaders are discussed in the next sub-sections.

5.1.2 Vertex Shader

The Vertex Shader is the programmable Shader stage in the rendering pipeline that handles the processing of individual vertices. The vertex shader usually performs traditional graphics operations, such as, Vertex transformation, Normal transformation and normalization, Texture coordinate generation, Texture coordinate transformation, and Lighting.

The vertex shader replaces the full functionality of the vertex processor. A vertex shader is responsible for replacing all the required functionality of this stage of the pipeline, which otherwise was taken care by fixed pipeline vertex transformation stage.

5.1.3 Fragment Shader

The Fragment shader is a programmable unit that operates on fragment values and their associated data. The fragment processor usually performs traditional graphics operations, such as, operations on interpolated values, Texture access, Texture application, and fog.

Fragments are per-pixel data structures that are created by the rasterization of graphics primitives. A fragment contains all the data necessary to update a single location in the frame buffer. Fragment processing consists of the operations that occur on a per-fragment basis, most notably reading from texture memory and applying the texture value at each fragment.

The inputs for this unit are the interpolated values computed in the previous stage of the pipeline such as vertex positions, colors, normals, etc. The fragment processor operates on single fragments, i.e. it has no clue about the neighboring fragments. One important point is that a fragment shader can not change the pixel coordinate, as computed previously in the pipeline. Vertex processor (using modelview and projection matrices) can be used to transform the vertex. The fragment shader has access to the pixels location on screen but it can not change it.

5.1.4 OpenGL Shading Language

GLSL is a high-level graphics programming language formally called the openGL shading langugage. With the OpenGL Shading Language, the fixed functionality stages for vertex processing and fragment processing have been augmented with programmable stages that can do everything the fixed functionality stages can do and a whole lot more. The OpenGL Shading Language allows application programmers to express the processing that occurs at those programmable points of the OpenGL pipeline.

In OpenGL Shading Language data flows from the application to the vertex processor, on to the fragment processor, and ultimately to the frame buffer. The OpenGL Shading Language was carefully designed to allow hardware implementations to perform parallel processing of both vertices and fragments. This parallel processes run on a hardware called Graphics Programming Unit(GPU).

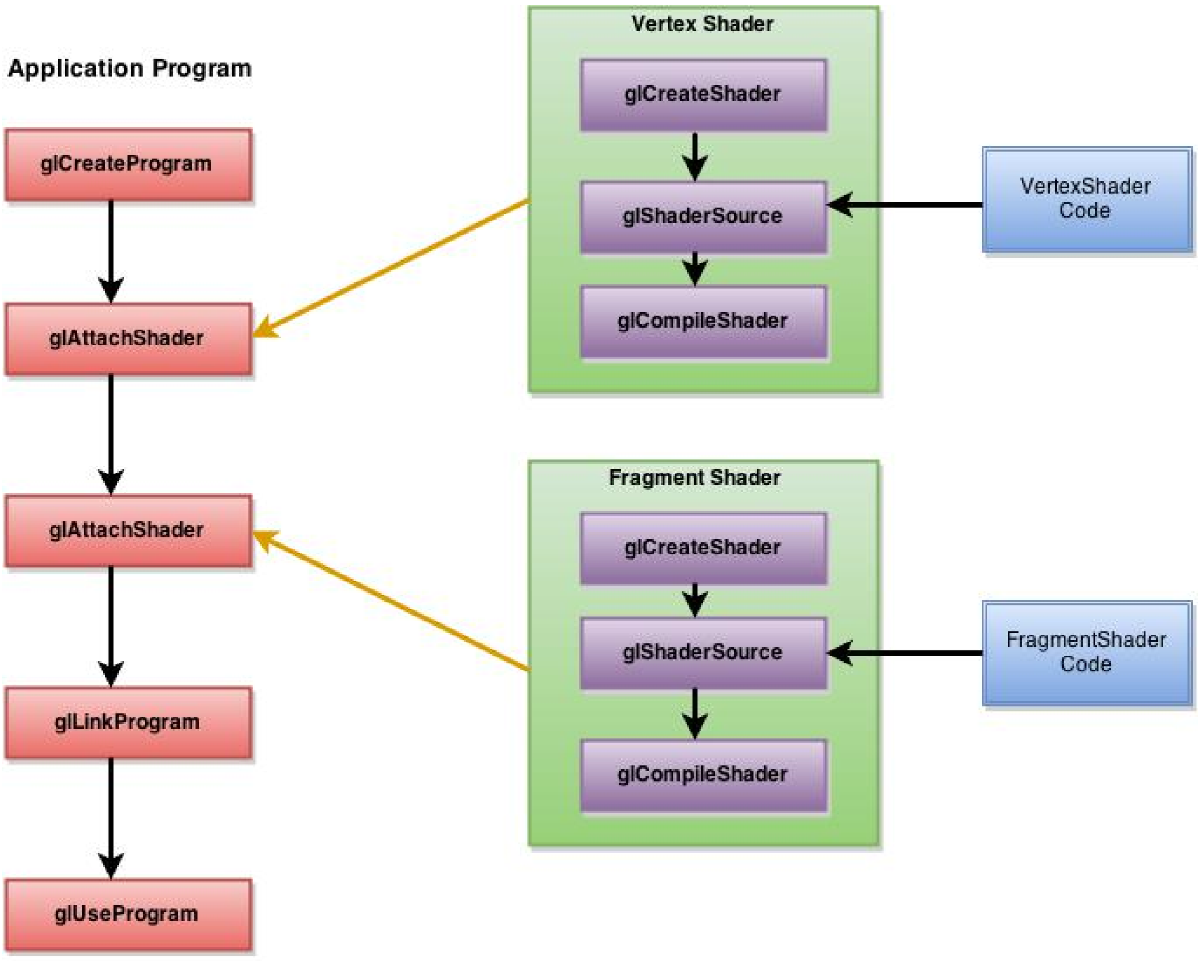


Figure 5.2: OpenGL GLSL program setup.

5.1.5 GLSL Setup For Execution

Every openGL Shading Language application, no matter how simple, must provide both a vertex shader and a fragment shader. Setting up the shaders requires a number of steps, including reading the shader code from files, compiling the code, and linking the shaders with the application. There is a set of openGL functions that deals with how to create vertex and shader objects, link them with an OpenGL applications, and enable the passing of variables among the OpenGL program and the shaders. As depicted by the figure 5.2, there are usually eight steps in initializing one or more shaders in the applications:

- Read the shader code
- Create a program object.
- Create shader object.
- Attach the shader object to the program object.
- Compiler the shaders.
- Link everything together.
- Select current program object.
- Align uniform and attribute variables between the application and the shaders.

5.2 Communication to GPU shaders

An application in OpenGL has several ways of communicating with the shaders, few are discussed below:

- **Uniform Buffer Objects** are global GLSL variables declared with the "uniform" storage qualifier. Uniforms are so named because they do not change from one execution of a shader program to the next within a particular rendering call.
- **Buffer Textures** is a one-dimensional texture whose storage comes from a Buffer Object. They are used to allow a shader to access a large table of memory that is managed by a buffer object.
- **Shader Storage Buffer Objects** They are a lot like uniform buffers, with their syntax almost identical to that of uniform buffer blocks. The principle difference is that you can write to them. One issue with them is that their minimum size is 16MB.

  The above techiques enables flow of data from application program to GPU shaders. But the only way to obtain output from shaders to the application program, is to render to some targets, usually the color and depth buffers.

5.3 Datasets

Datasets used were taken from www.volvis.org, which is publicly available. All datasets used are in RAW binary format. For multimodal visualization, we have used a fabricated PET dataset.

Table 5.1: Dataset details

Dataset	Dimension
Foot	256x256x256x1
Stent	512x174x512x1
head	256x256x256x1

5.4 Two-pass Rendering

In the implementation of two pass rendering raycasting which is discussed in section 2.3.2, we make use of two pair of vertex and fragment shaders. In first pass, the first pair of shaders writes color location of cube with the front face culled, onto a 2D RGB texture. The location of the vertex is also its color which is assigned to the cube in the application program. In this first pass, the output of vertex shader are the transformed and colored vertices of the cube, assembled into primitives and fed into rasterizer, which then interpolates the color values between the vertices. The fragments generated by rasterizer are sent to fragment shader, which in turn writes to a 2D RGB texture. This method of capturing images that would normally be drawn to the screen onto a 2D texture is called *offscreen rendering*. This 2D RGB texture holds the clip-space coordinate which is the exit point of the ray originated from the camera.

In the second pass, the second shader pair are used to generate the final image which is rendered on a window. In this pass, 3D texture is used to store the volume data and it is sent across to shaders as uniform objects. We use texture samplers to retrieve the voxel data from the 3D texture. Along with this, the 2D RGB texture which holds the bounding box exit point, is bound as uniform object and is used in the fragment shader. In this pass, the backface of the bounding box is culled. In the vertex shader, vertices are transformed and colored similar to first pass. The fragment shader receives fragments color from the rasterizer, this corresponds to the entry location of the ray on the bounding box. Now, we have both entry point and exit point which is obtained using 2D RGB texture. After computing the direction of the ray using entry and exit point, we start from the entry location and move towards exit location sampling values at uniform location defined by the stepsize. We use texture sampler to fetch data values from the 3D texture for the given location. The values received from the texture sampler is used for front-to-back compositing as discussed in the section 2.3.4.2.

When compositing, we also compute the histogram as discussed in the section 3.4. The histogram computed for each ray is written to an offscreen buffer. At the application program, by summation of all texel values in the buffer we obtain the final histogram and it is displayed on a seperate window.

5.5 Graphics pipeline in hardware

The graphics pipeline is well suited to the rendering process because it allows the GPU to function as a stream processor since all vertices and fragments can be thought of as independent. This allows all stages of the pipeline to be used simultaneously for different vertices or fragments as they work their way through the pipe. In addition to pipelining vertices and fragments, their independence allows graphics processors to use parallel processing units to process multiple vertices or fragments in a single stage of the pipeline at the same time.

In this thesis, we use ray casting to perform all the computations involved in visibility-guided volume rendering: computing visibility histograms and compositing for direct volume rendering. In both of these, the computations per ray are independent of other rays. Hence, the algorithm is intrinsically parallelizable and has been programmed on the GPU by implementing vertex and fragment shader as described in section 5.4. This resulted in performance that supports interactivity.

Chapter 6

Results

We applied our approach dicussed in the thesis to a number of datasets, both medical and non-medical. The use of visibility histogram alone prove to be an important element towards the understanding of complex datasets. Results show that visibility histogram are a useful aid to quantify the information derived from the structure of interest in both single and multimodality setup. The technique discussed helps user generate desired visualization, by intuitively designing the transfer function, and thus enabling user to gain insights into the volume. We have implemented it on GPU shaders, leading to improved performance with better user experience.

As part of future work, we can incorporate lighting for better illustrative visualization. We can extend to two-dimensional transfer functions, for silhouette rendering in ROI and produce enhanced visualizations by incorporating NPR techniques. Volume raycasting and visibility driven transfer function techniques can also be used to visualize iso-surfaces of volumetric dataset. Another important future work is to extent our work to enable visualization of non-uniform grids. We can also extend our approach to study effects of intergrating other NPR technique for multimodal visualization. We can work on the implementational aspects of the semi-automatic transfer function generation.

Bibliography

[1] R. C. Barthold Lichtenbelt and S. Naqvi. *Introduction to Volume Rendering.* Prentice-Hall, fifth edition, 1998.

[2] S. Bruckner and M. E. Gr"oller. Exploded views for volume data. *IEEE Transactions on Visualization and Computer Graphics*, 12(5):1077–1084, 9 2006.

[3] W. Cai and G. Sakas. Data intermixing and multi-volume rendering. *Comput. Graph. Forum*, 18(3):359–368, 1999.

[4] C. Correa and K.-L. Ma. Size-based transfer functions: A new volume exploration technique. *Visualization and Computer Graphics, IEEE Transactions on*, 14(6):1380–1387, Nov 2008.

[5] C. Correa and K.-L. Ma. Visibility-driven transfer functions. In *Visualization Symposium, 2009. PacificVis '09. IEEE Pacific*, pages 177–184, April 2009.

[6] C. D. Correa, D. Silver, and M. Chen. Discontinuous Displacement Mapping for Volume Graphics. In R. Machiraju and T. Moeller, editors, *Volume Graphics.* The Eurographics Association, 2006.

[7] F. de Moura Pinto and C. M. D. S. Freitas. Importance-aware composition for illustrative volume rendering. In *SIBGRAPI 2010, Proceedings of the 23rd SIBGRAPI Conference on Graphics, Patterns and Images, Gramado, Brazil, August 30 2010-September 3, 2010*, pages 134–141, 2010.

[8] D. Ebert and P. Rheingans. Volume illustration: Non-photorealistic rendering of volume models. In *Proceedings of the Conference on Visualization '00*, VIS '00, pages 195–202, Los Alamitos, CA, USA, 2000. IEEE Computer Society Press.

[9] T. T. Elvins. A survey of algorithms for volume visualization. *SIGGRAPH Comput. Graph.*, 26(3):194–201, Aug. 1992.

[10] A. Hertzmann. Non-photorealistic rendering and the science of art. In *Proceedings of the 8th International Symposium on Non-Photorealistic Animation and Rendering*, NPAR '10, pages 147–157, New York, NY, USA, 2010. ACM.

[11] J. Kruger, J. Schneider, and R. Westermann. Clearview: An interactive context preserving hotspot visualization technique. *Visualization and Computer Graphics, IEEE Transactions on*, 12(5):941–948, Sept 2006.

[12] M. Levoy. Display of surfaces from volume data. *IEEE Comput. Graph. Appl.*, 8(3):29–37, May 1988.

[13] W. Li, L. Ritter, M. Agrawala, B. Curless, and D. Salesin. Interactive cutaway illustrations of complex 3d models. *ACM Trans. Graph.*, 26(3):31, 2007.

[14] I. Manssour, S. Furuie, L. Nedel, and C. Freitas. A multimodal visualization framework for medical data. In *Computer Graphics and Image Processing, 2000. Proceedings XIII Brazilian Symposium on*, pages 356–, 2000.

[15] M. J. McGuffin, L. Tancau, and R. Balakrishnan. Using deformations for browsing volumetric data. In *14th IEEE Visualization 2003 Conference (VIS 2003), 19-24 October 2003, Seattle, WA, USA*, pages 401–408, 2003.

[16] R. S. Novelline. *Fundamentals of Radiology.* Society for Industrial and Applied Mathematics, fifth edition, 1997.

[17] A. Patel and K. Mehta. 3d modeling and rendering of 2d medical image. In *Communication Systems and Network Technologies (CSNT), 2012 International Conference on*, pages 149–152, May 2012.

[18] L. C. Robert A. Drebin and P. Hanrahan. Volume rendering. *In Proceedings of the 15th annual conference on Computer graphics and interactive techniques.*

[19] Y. Sato, C.-F. Westin, A. Bhalerao, S. Nakajima, N. Shiraga, S. Tamura, and R. Kikinis. Tissue classification based on 3D local intensity structure for volume rendering. *IEEE Trans on Visualization and Computer Graphics*, 6(2):160–180, 2000.

[20] R. Spence. *Information Visualization.* ACM Press, first edition, 2001.

[21] I. Viola, A. Kanitsar, and M. E. Gr"oller. Importance-driven feature enhancement in volume visualization. *IEEE Transactions on Visualization and Computer Graphics*, 11(4):408–418, 2005.

[22] L. Wang, Y. Zhao, K. Mueller, and A. Kaufman. The magic volume lens: an interactive focus+context technique for volume rendering. In *Visualization, 2005. VIS 05. IEEE*, pages 367–374, Oct 2005.

[23] D. Weiskopf, K. Engel, and T. Ertl. Interactive clipping techniques for texture-based volume visualization and volume shading. *Visualization and Computer Graphics, IEEE Transactions on*, 9(3):298–312, July 2003.

[24] C. M. Wittenbrink, T. Malzbender, and M. E. Goss. Opacity-weighted color interpolation for volume sampling. In *IEEE Symposium on Volume Visualization*, pages 135–142, 1998.

[25] L. Zheng, C. D. Correa, and K.-L. Ma. Visibility guided multimodal volume visualization. In *In Proceedings of Bioinformatics and Biomedicine (BIBM), 2013 IEEE International Conference*, pages 297–304, 12 2013.

[26] Z. Zhou, G. Wang, C. Huang, and H. Lin. Opacity modulation based peeling for direct volume rendering. In *Industrial Electronics and Applications (ICIEA), 2014 IEEE 9th Conference on*, pages 1152–1157, June 2014.