UNIVERSITY OF CALIFORNIA, SAN DIEGO

Visual Exploration in Volume Rendering for Multi-Channel Data

A dissertation submitted in partial satisfaction of the
requirements for the degree
Doctor of Philosophy

in

Computer Science

by

Han Suk Kim

Committee in charge:

Jürgen P. Schulze, Chair
Henrik Wann Jensen
Falko Kuester
Maryann E. Martone
Larry Smarr

2011

The dissertation of Han Suk Kim is approved, and it is acceptable in quality and form for publication on microfilm and electronically:

_____

_____

_____

_____

_____

Chair

University of California, San Diego

2011

DEDICATION

To my dear grandmother.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGEMENTS

First of all, I would like to thank my advisor, Jürgen Schulze, for his endless help and support throughout my studies. He has spent many hours discussing research direction, expounding on new ideas, and even reviewing my code. Without his guidance, it would not have been possible for me to come this far. I am also grateful to my committee members: Larry Smarr, Henrik Wann Jensen, Maryann Martone, and Falko Kuester. Larry Smarr and Falko Kuester have given me a lot of great advice on my research direction. Henrik Wann Jensen took time to discuss the details of my research projects. Maryann Martone has been very generous in supporting my research through the National Center for Microscopy and Imaging Research and in introducing me to many great scientists in the center.

The past six years have brought many ups and downs. The great benefit of this adventure was that I was able to work with many great people from whom I could learn a lot. I would like to thank Andrew Chien, my first advisor, who gave me an opportunity to start this journey. Members of his research group, CSAG, especially Ryo Sugihara, Richard Huang, Dionysios Logothetis, Yangsuk Kee, and Jerry Chou, gave me much help in figuring out many things in graduate life and my studies.

I was fortunate to work with Scott Baden, my second advisor. I was able to explore many exciting projects in scientific computing. I would like to thank his continuing support and advice on my research, and many parts of my research would not have been possible if he had not allowed me to use the computing resources in his lab. I would also like to thank his lab members, Didem Unat, with whom I collaborated, and Pietro Cicotti and Jacob Sorensen, with whom I spent many hours discussing research topics.

Besides my accommodating advisors, I was very fortunate to work with many great people at the California Institute of Telecommunications and Information Technology. Researchers in the GEON project, Chaitan Baru, Ramon Arrowsmith, and Christopher Crosby, supported the earlier stage of my research. I would also like to thank Mark Ellisman and Steve Peltier for their long support of my research through NCMIR. I would not have been able to include most figures in this dissertation without the following people's help: Anglea Cone, Gina Sosinsky, Eric Bushong, Masako Terada, and Raj

Singh. The Whole Brain Catalog project team members, Stephen Larson, Chris Aprea, and David Little, gave me a great chance to work on the exciting project.

It would have taken longer for me to settle in visualization research without the help of members of the Immersive Visualization Laboratory and UCSD Graphics Lab, in particular: Philip Weber, Andrew Prudhomme, Shaofeng Liu, Mabel Mengzi Zhang, Gregory Long, and Kuei-Chun Hsu in IVL, and Iman Sadeghi, Toshiya Hachisuka, Oleg Bisker, and Carlos Dominguez Caballero, Matthias Zwicker in the graphics lab.

Throughout my studies, I had two exciting excursions. I would like to thank Daniel Quinlan for taking me as a summer intern at the Lawrence Livermore National Laboratory, and John Wilkes and Sunita Verma for allowing me to explore information visualization research at Google Inc. Through the internships, I met many excellent researchers and interns and made many fond memories with them.

I am grateful to all of my great friends here at UCSD and other parts of the world for sharing my concerns, joy, and stress for the past six years. I particularly thank Donghwan Jeon, Peter Shin, Youngmin Cho, Dokyum Kim, Sangtae Kim, and Katherine Cordova in San Diego, and Soonmin Ko, Changhan Rhee, Jeechul Woo, Ikkjin Ahn, and Sungjun Ahn in the distance.

I would like to thank my family for their endless support and love. It is a rather long list, but Doochul Kim, Heasook Rhee, Keejung Han, Michael Price, Yunie Kim, Joonseok Kim, and Peter, Heasoon, and Nadia Arzberger have shown endless love and care, and I am deeply indebted to every one of them for their encouragement and belief.

Lastly, but not least, my late grandmother has taught me a lot of important values of life: loving and sacrificing for family, being selfless, and working hard. I owe her too much to describe by word. I hope that she is also pleased with my accomplishment.

Portions of this dissertation are based on papers which I have co-authored with others. My contributions to each of these papers are listed below.

- Chapter 4 and Chapter 5 are based on material published in the articles:

  Han Suk Kim, Jürgen P. Schulze, Angela C. Cone, Gina E. Sosinsky, Maryann E. Martone: Dimensionality reduction on multi-dimensional transfer functions for multi-channel volume data sets. Information Visualization, Vol. 9 No. 3: pp. 167-180 (2010). Palgrave Macmillan.

Han Suk Kim, Jürgen P. Schulze, Angela C. Cone, Gina E. Sosinsky, Maryann E. Martone: Multichannel transfer function with dimensionality reduction. VDA 2010: SPIE Proceedings 75300.

I was the primary investigator and author of these papers.

- Chapter 6 and Chapter 7 are based on material that is in preparation for submission:

  Han Suk Kim, Didem Unat, Scott B. Baden, Jürgen P. Schulze, "Real-Time Data-Centric Viewpoint Selection."

  I was the primary investigator and author of this paper.

VITA

2005	Bachelor of Science, Seoul National University
2008	Master of Science, University of California, San Diego
2011	Doctor of Philosophy, University of California, San Diego

PUBLICATIONS

Han Suk Kim, Didem Unat, Scott B. Baden, Jürgen P. Schulze, "Real-Time Data-Centric Viewpoint Selection." (in preparation)

C. J. Crosby, S. Krishnana, J R. Arrowsmith, H. S. Kim, J. Colunga, N. Alex, C. Baru, "Points2Grid: An efficient local gridding method for DEM generation from Lidar point cloud data," Geosphere, 2011. (under review)

Didem Unat, Han Suk Kim, Jürgen P. Schulze, Scott B. Baden, "Auto-optimization of a feature selection algorithm," In *Proceedings of the 4th Workshop on Emerging Applications and Many-core Architecture*, June 2011.

Han Suk Kim, Sunita Verma, John Wilkes, "Interactive visual exploration of service level objectives," Technical Report CS2011-0966, University of California San Diego, May 2011.

Jürgen P. Schulze, Han Suk Kim, Philip Weber, Andrew Prudhomme, Roger E. Bohn, Maurizio Seracini, Thomas A. Defanti, "Advanced Applications of Virtual Reality," Advances in Computers 82: 217-260 (2011).

Larson SD, Aprea C, Martinez J, Little D, Astakhov V, Kim HS, Zaslavsky I, Martone ME, Ellisman MH, (2010) An Open Google Earth for Neuroinformatics: The Whole Brain Catalog, Neuroinformatics 2010, Kobe Japan. (Spotlight demo presentation)

Han Suk Kim, Jürgen P. Schulze, Angela C. Cone, Gina E. Sosinsky, Maryann E. Martone: Dimensionality reduction on multi-dimensional transfer functions for multi-channel volume data sets. Information Visualization Vol. 9 No. 3: pp. 167-180 (2010). Palgrave Macmillan.

Han Suk Kim, Jürgen P. Schulze, Angela C. Cone, Gina E. Sosinsky, Maryann E. Martone: Multichannel transfer function with dimensionality reduction. VDA 2010: SPIE Proceedings 75300.

Han Suk Kim, Jürgen P. Schulze, "High resolution video playback in immersive virtual environments," UCSD Technical Report CS2009-0952, Nov 2009.

Kim, H.S., and Schulze, J., High Resolution Video Playback in Immersive Virtual Environments, IEEE Virtual Reality 2009 Conference, 2009.

Crosby, C.J., Arrowsmith, J R., Frank, E., Nandigam, V., Kim, H.S., Conner, J., Memon, A., Baru, C., Enabling Access to High-Resolution LiDAR Topography through Cyberinfrastructure-Based Data Distribution and Processing, The Annual Meeting of the American Association of Geographers, Abstract of 103rd Annual Meeting, San Francisco, CA, 2007.

Kim, H., Arrowsmith, J R., Crosby, C.J., Jaeger-Frank, E., Nandigam, V., Memon, A., Conner, J., Baden, S.B., Baru, C., An Efficient Implementation of a Local Binning Algorithm for Digital Elevation Model Generation of LiDAR/ALSM Dataset, Eos Trans. AGU, 87(52), Fall Meet. Suppl., Abstract G53C-0921, 2006.

Crosby, C.J., Arrowsmith, J R., Frank, E., Nandigam, V., Kim, H.S., Conner, J., Memon, A., Baru, C., Enhanced Access to High-Resolution LiDAR Topography through Cyberinfrastructure Based Data Distribution and Processing, Eos Trans. AGU, 87(52), Fall Meet. Suppl., Abstract IN41C-04, 2006.

ABSTRACT OF THE DISSERTATION

Visual Exploration in Volume Rendering for Multi-Channel Data

by

Han Suk Kim

Doctor of Philosophy in Computer Science

University of California, San Diego, 2011

Jürgen P. Schulze, Chair

Volume rendering has been an important tool to understand scientific 3D data. Traditional volume data contain only one value per voxel, but light microscopy captures multiple protein expressions from different fluorescences. This technology generates multi-channel data where several values are defined at each voxel. Because traditional volume rendering systems have assumed single channel information, i.e., opacity, there exists a significant gap between the technology available for single channel and for multi-channel data, especially in visual exploration methods for better understanding of the data.

In this dissertation, we bridge the gap by investigating the characteristics of multi-channel data. The visual exploration in volume rendering has been done in many

ways, but two of the most critical and effective approaches for multi-channel data are transfer function design and viewpoint selection.

We first propose a new method for multi-channel transfer function design. The challenge for designing multi-dimensional transfer functions is the dimensionality of the domain. Multi-channel data often contain more than three values per voxel, which prevents users from manipulating color and opacity on multi-dimensional domain. Moreover, adding additional attributes, such as gradient, second order derivatives, or textural information, further increases the dimension of the domain. We apply recently-developed nonlinear dimensionality reduction algorithms to reduce the high dimensionality of the domain. In this work, we use Isomap and Locally Linear Embedding as well as Principal Component Analysis.

Furthermore, we present a real-time viewpoint selection algorithm for multi-channel data. Because the transfer function dramatically changes the appearance of the multi-channel data, users see different objects in the data depending on the transfer function exploration. This characteristic of visualization of multi-channel data necessitates real-time viewpoint selection. The automatic viewpoint selection in the course of transfer function exploration enables users to quickly understand the data. Our algorithm takes under a second for various volume data sets, which is about 40 to 80 times faster than in previous approaches. This allows the algorithm to be integrated with real-time transfer function exploration.

Chapter 1

Introduction

Volume rendering has been an important tool to explore and navigate 3D scientific data [DCH88]. As computing capacity exponentially increases and as the technology of acquiring biological samples develops, one can easily encounter 3D data sets in many scientifc disciplines [EYSK94, Zeh06, CLB05]. In order to help scientists explore their data interactively and understand their data effectively, computer visualization communities have worked on developing volume rendering systems since the late 1980s [Lev88]. Although now we have volume rendering systems that can render many scientific data sets with an impressive quality, there is still room for improvement in interactively exploring data sets from neuro-biology research, called multi-channel data.

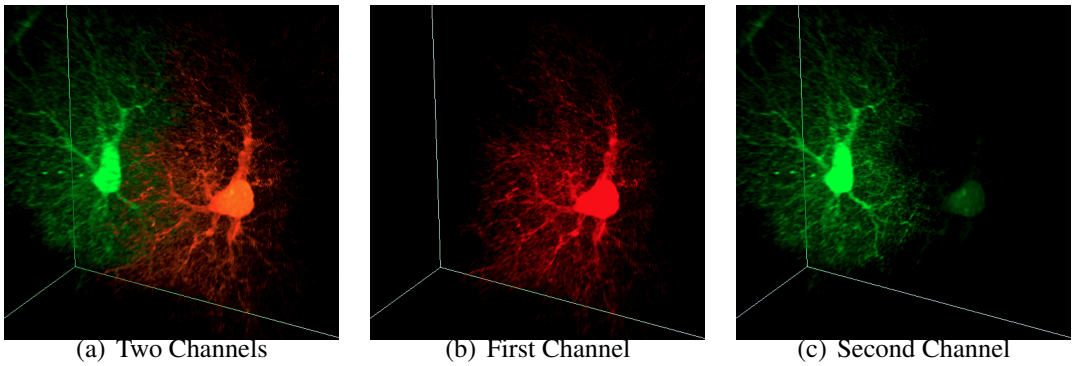
This dissertation investigates new methods of improving user experience when exploring multi-channel data with volume rendering. There exists a significant gap between the technologies available for multi-channel data and single-channel data, especially in visually exploring the data. We propose a solution to narrow the gap. Furthermore, we present an improved algorithm to enhance the visual interaction when exploring the volume data.

1.1 Multi-Channel Data in Neuro-Biology Research

In a confocal laser scanning microscope [Paw89a], a laser beam is focused onto a small spot of a specimen. The focal point can be either inside or on the surface of

the specimen. By moving the focal length up and down, the microscope can optically section the specimen, leaving the sample physically intact. The amount of light emitted from the focal point is proportional to the light emission in that area. The specimen is dyed with different fluorescences, each of which emits a different wavelength. The image from each wavelength is stored as a *channel* and in neuro-biological research, one specimen is sampled multiple times with different flourescences, which creates multi-channel data.

Figure 1.1 shows an example of multi-channel data. This data has two channels and each fluorescence is colored in red and green, respectively. Figure 1.1(b) and Figure 1.1(c) show the rendered images of each channel, while Figure 1.1(a) shows the combined channels. Because each channel shows different proteins, what we see in each channel is dramatically different.



(a) Two Channels (b) First Channel (c) Second Channel

Figure 1.1: Volume rendered images of two adjacent protoplasmic astrocytes. The first channel is colored in red and the second channel is colored in green.

1.2 Motivation

While single channel data has been extensively studied by many visualization communities, multi-channel data has emerged recently and has distinct characteristics separating it from single channel data [FDM+00]. Although there are several volume rendering systems for multi-channel data, specifically designed for neuro-biology [Vis11, BIT11, Per11, WOCH09], we see that the systems lack the capability of advanced visual

exploration. We investigate how we can improve user experience when users interactively explore volume data sets.

1.2.1 Transfer Function

Since many problems in "visualizing 3D volumes" are commonly considered to be solved [WWHZ00, EKE01, BD02, GWGS02, PTCF02], scientists started to look into the problem of "exploring 3D volumes" [KD98, PLS+00, Kin02, PF08]. The exploration includes modifying color and opacity in real time so that one can gain more insight about the data, focus on a particular part of data, or highlight areas of interest interactively. This type of real-time modification is an essential tool, which is done by transfer function design [KKH02], to obtain rendered images of the best quality [PLS+00, Kin02]. The most common way of exploring color and opacity with transfer functions is to put the rendering panel and transfer function panel side by side, and to manually define the function [KKH01]. Although this trial-and-error process may take a long time, one can gain deeper knowledge about the data during the process [PLS+00].

However, the transfer function for multi-channel data has not been studied extensively. This is mainly due to the amount of information in the data. Multi-channel data itself contains multiple values per voxel, and an advanced transfer function, called a *multi-dimensional transfer function*, requires additional information to locate areas of interest in the data. A multi-dimensional transfer function for single channel data is defined on a multi-dimensional histogram, consisting of the original intensity and additional information (e.g. gradient magnitude [KD98] or curvature [KWTM03]). Applying the idea of multi-dimensional transfer function to multi-channel data is challenging because adding the derived values to the set of intensities causes the histogram domain to exceed what human vision can perceive, namely 3D space.

1.2.2 Viewpoint Selection

Another way of "exploring 3D volumes" is done by basic transformations of volume: i.e., rotation, zooming in and out, or translation. Among these essential transformations, rotating the volume data to a different angle has the biggest impact on what

we see on the screen [BTB98]. Users of volume rendering systems usually rotate the volume to find out whether or not the current transfer function effectively shows areas of interest. This exploration is a time-consuming iterative process: users modify the transfer function, rotate the volume to check the appearance, and continue updating the transfer function.

Viewpoint selection is an automatic method that finds the best viewing direction from which users can see and understand the entire volume effectively. By automatically positioning the camera at the best location, users do not need to manually rotate the volume to check their transfer functions. Therefore, a viewpoint selection algorithm can improve user experience during the transfer function exploration. The view selection algorithm, however, must run fast enough to be incorporated into the transfer function exploration. Previous approaches [BS05, TFTN05, JS06, TLB+09] have failed to meet this requirement because most viewpoint selection algorithms are based on exhaustive searching.

1.3 Our Contributions

This dissertation investigates two ways of effectively exploring multi-channel 3D volume data: transfer function and automatic viewpoint selection. The key contributions are methods and technologies for visually exploring multi-channel data. We outline our major contributions below.

1.3.1 Transfer Function

Multiple Features: We propose a new framework in which multiple features of multiple channels can be integrated in the transfer function domain. Traditional approaches have restricted the domain to 2D or 3D because it is otherwise difficult to navigate. We show that our method can reduce a 24D feature domain to 3D with 23% loss of total variance, and it can reduce a 6D feature domain to 3D with only 6% loss of total variance. This reduced domain allows users to easily find areas of interest in multi-channel volume data.

Multiple Channels: Our new method opens up a new way of visually exploring multi-channel data. Our method enables users to examine data where two or more features have positive and/or negative correlations. Many scientific data sets have multi-variate information. Examining multiple fields at the same time, rather than visualizing one field at a time, allows users to find areas of interest more efficiently.

Dimensionality Reduction: Recent research on non-linear dimensionality reduction algorithms has shown that the proposed algorithms can successfully find a low dimensional representation of very high-dimensional data. We apply this idea to the transfer function domain. Our work shows the effectiveness of dimensionality reduction algorithms in a field to which they have not previously been applied.

1.3.2 Viewpoint Selection

Real-Time Algorithm: We propose a new method for viewpoint selection. The algorithm runs in real-time (under one second for most data), which provides an interactive viewpoint update with transfer function exploration. Our algorithm extends a feature detection algorithm to 3D data and formulates the viewpoint selection as an optimization problem. Compared to previous approaches, the simplicity of our algorithm greatly benefits the performance. Our algorithm outperforms previous approaches by a factor of 27 to 38, on average, while achieving comparable qualities in resulting images.

User Experience: Transfer functions can change the appearance of the volume significantly because users can interactively define opacity. In the case of multi-channel data, it is much more likely that viewers see completely different parts of the data in the course of transfer function exploration. Our real-time viewpoint selection algorithm helps viewers to quickly understand the current setting of the transfer function by interactively rotating the volume to the optimal viewpoint.

Computer Vision: The majority of feature selection algorithms in computer vision research focus on 2D images. Our work shows that it is possible to extend a feature

selection algorithm to 3D volume data. The main challenge for extending the feature selection algorithm is the computational overhead due to the sheer amount of volume data in comparison to that of 2D images. We optimized this computationally expensive algorithm with a heterogeneous computing architecture.

1.4 Organization of the Dissertation

This dissertation is organized into eight chapters. Beginning in Chapter 2, we discuss the characteristics of multi-channel data. In Chapter 3, we review the basic theory of volume rendering and the previous efforts in volume rendering for multi-channel data. We then describe the role of transfer functions in volume rendering and discuss the lack of transfer function support for multi-channel data in Chapter 4. In Chapter 5, we propose a new transfer function design method for multi-channel data. Chapter 6 discusses both the importance and limitations of viewpoint selection algorithms in transfer function exploration. Chapter 7 proposes our new viewpoint selection algorithm that greatly enhances the user experience in transfer function exploration. Finally, we summarize the major contributions of this dissertation, and discuss future work in Chapter 8.

Chapter 2

Multi-Channel Data

In this chapter, we define multi-channel data and discuss the characteristics of multi-channel data. The difference between single- and multi-channel data plays an important role in this dissertation.

2.1 Data Model

The data used in direct volume rendering is organized in a structured grid. Data points in a grid can be accessed with their index, x, y, and z. Figure 2.1 shows a schematic shape of a grid. The entire volume is divided into many small cubes, and each cube contains a value. This data structure can be easily mapped to a 3D array, which many program languages provide as a default data storage type.

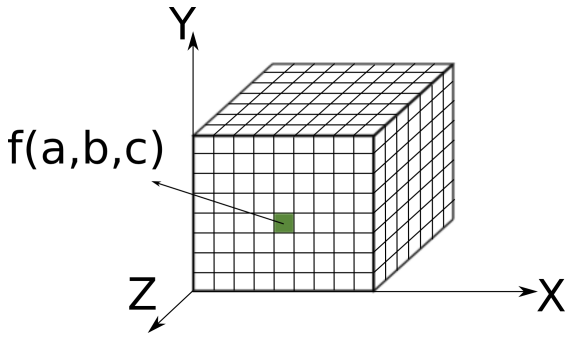


Figure 2.1: The data model for direct volume rendering. Data samples are stored in a structured grid. At each point (x,y,z), a value is defined as $f(a,b,c) \in \mathbb{R}$.

This grid can be considered as a set of regular samples from a function $f : \mathbb{R}^3 \rightarrow \mathbb{R}$ defined in 3D volume space. Because f returns a single value, we refer to this data as *single-channel* data. If the number of samples in the grid is large, that means the distance between two adjacent samples is small. If the distance increases, i.e., coarse resolution, it may not be possible to reconstruct the original volume, by the Nyquist sampling theorem [Jer77]. Volume rendering processes also interpolate those sample points to reconstruct the values on the points inside the volume, not necessarily aligned with the sample coordinates. This will be further discussed in Chapter 3.

There are many examples in scientific research where data is represented in this format. Computed Tomography (CT) [Her10] takes a stack of cross-section images of the human body or of an object, and the stack forms a 3D volume. Computation fluid dynamics researchers solve the Navier-Stokes equation with a finite difference method where a 3D structured grid is overlaid with the simulation spatial domain [MM05].

2.2 Definition of Multi-Channel Data

The model in Figure 2.1 assumes the function f returns a single scalar value. However, it is often desirable to have multiple values at each sample point. If multiple values are defined in the volume, we call this data *multi-channel* data. This type of data is especially common in neuro-biological research where different proteins of the specimen are stained with different colors. Each color, usually red, green, or blue, represents one channel. Figure 2.2 shows an example of multi-channel data.

The image is obtained by a confocal laser scanning microscope [Paw89b] at the National Center for Microscopy and Imaging Research (NCMIR) [Nat]. In a confocal laser scanning microscope, a laser beam is focused onto a small spot of a specimen, which will be a sample point at (x, y, z) as discussed in Section 2.1. The focus can be both inside or on the surface of the specimen. By moving the focal length up and down, the microscope can optically section the specimen, leaving the sample physically intact. The amount of light measured from the focal point is the value in the data, i.e., $f(x, y, z)$, and it is proportional to the light emission from that area. The specimen is stained with different fluorescences, each of which emits a different wavelength, and the image

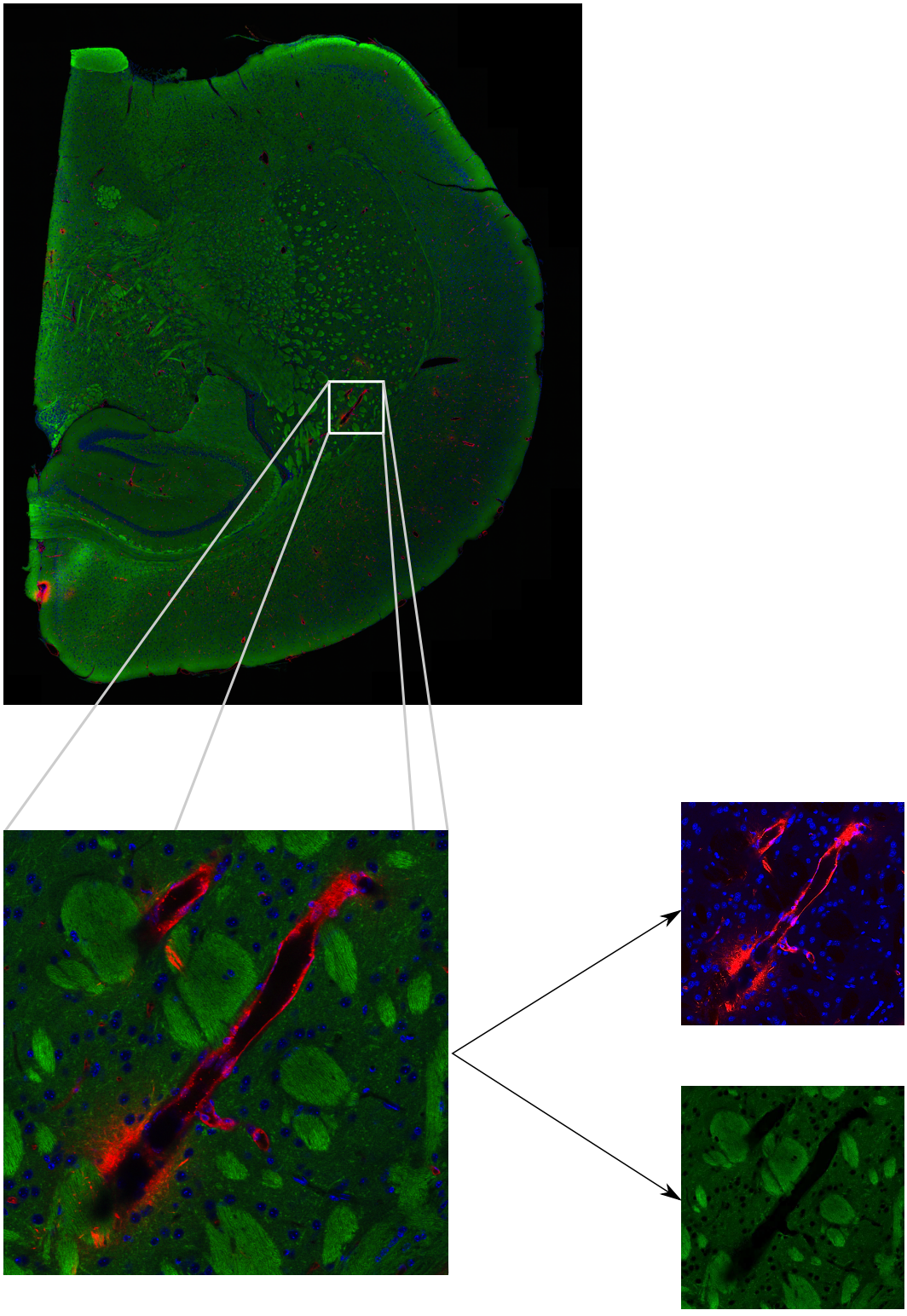


Figure 2.2: An example of multi-channel data. The image is obtained by a confocal laser scanning microscope at the National Center for Microscopy and Imaging Research.

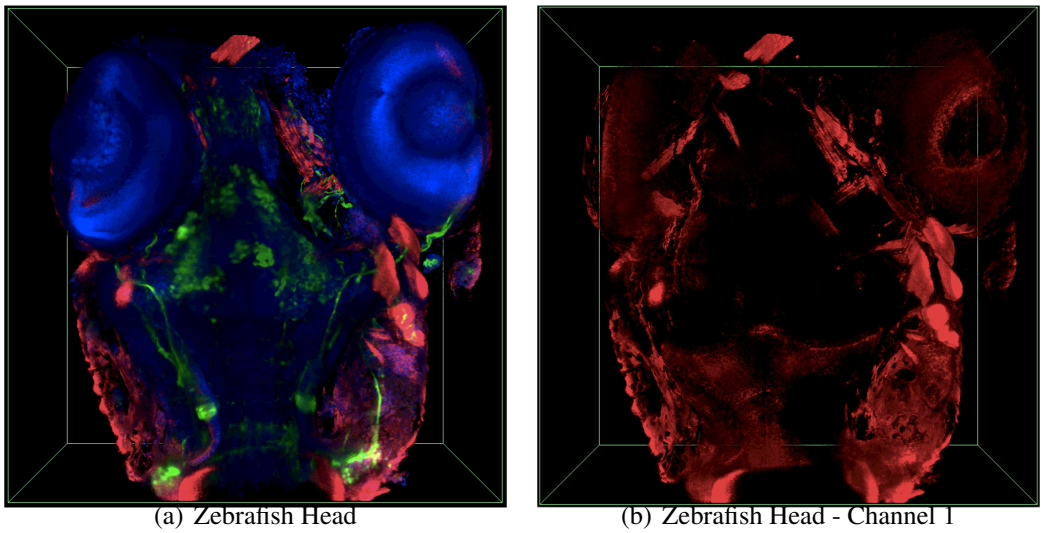
from each wavelength is stored as one *channel*. In neuro-biological research, scientists need to examine three or more channels to understand the specimen. There are three channels in this example, each of which labels myelin (green), blood vessels (red), and cells (blue). In order to label them, the tissue was stained with Fluoromyelin, Dil, and TO-PRO3. The microscope scans only a small region to get a small patch of the image, and once all the patches are acquired, they are stitched together to create one slice of size 8933 x 10823 pixels. This technique allows scientists to see the entire coronal mice brain tissue.

The bottom image in Figure 2.2 shows a small portion of the entire slice. In the image, blood vessels (colored in red) are shown diagonally. An important characteristic of this type of multi-channel data is, as shown on the bottom right-hand side of Figure 2.2, that the objects shown in each channel are dramatically different from each other. Cells in the blue channel are more sparse and spreads out across the image, whereas blood vessels in the red channel appear to be solid objects. The green channel fills up the rest of the image.

Another example is shown in Figure 2.3. The data shows a five-day-old zebrafish head, and is courtesy of Hideo Otsuna in the Department of Neurobiology and Anatomy, and Yong Wan in the Scientific and Imaging Institute, both at the University of Utah. The multi-channel data has three channels and is rendered with a 3D texture slicing method. The most interesting aspect of multi-channel data is, as we can see in this figure, that each channel shows different parts of the specimen. The red channel shows the muscles, the green channel highlights the neurons, and the blue channel visualizes the nuclei. Although each of the three channels is taken from the same spatial location, the points of interest are all different because the rendered images are radically different.

2.3 Summary

In this chapter, we defined multi-channel data and discussed the characteristics of multi-channel data. Multi-channel data contains multiple values at each sample point, each of which plays an important role in scientific research. Because each channel from a confocal light scanning microscope expresses different protein structures, each



(a) Zebrafish Head (b) Zebrafish Head - Channel 1

(c) Zebrafish Head - Channel 2 (d) Zebrafish Head - Channel 3

Figure 2.3: Volume rendered image of five-day-old zebrafish head. The 3D multi-channel data is courtesy of Hideo Otsuna and Yong Wan at the University of Utah. The data contains three channels, colored in red (muscles), green (neurons), and blue (nuclei). Different channels show different aspects of the original specimen.

channel visualizes dramatically different objects, and points of interests also change depending on which channel we are to visualize.

Chapter 3

Direct Volume Rendering

As we have discussed in the previous chapter, multi-channel data is organized as a 3D structured grid. The 3D grid data is much more dense than mesh or isosurface data because each grid point has multiple values. Direct volume rendering has proved itself as an effective tool for such dense data because each voxel data contributes to the final image in the rendering. In this chapter, we first review the background theory of the volume rendering algorithms. We further discuss previous approaches in volume rendering for multi-channel data.

3.1 Rendering Equation

Direct volume rendering, often called volume rendering, is an evaluation of a rendering equation that specifies how the light, or energy, is emitted and absorbed when a ray passes through the 3D volume.

In order to define the rendering equation, we first model how the light reaches the viewer's eye. Figure 3.1 shows the volume rendering model. A volume is thought of as a cloud filled with particles emitting and absorbing light. The energy of wavelength λ at location $s \in \mathbb{R}^3$ is described by $C_\lambda(s)$, and the particle density at point s is defined as $\mu(s)$. Let $I_\lambda(\mathbf{x}, \mathbf{r})$ be the intensity of wavelength λ at location $\mathbf{x}$ on the image plane coming from ray direction $\mathbf{r}$. The intensity I_λ is calculated by evaluating the following integral [Kru90], [Max95], [MHB+00].

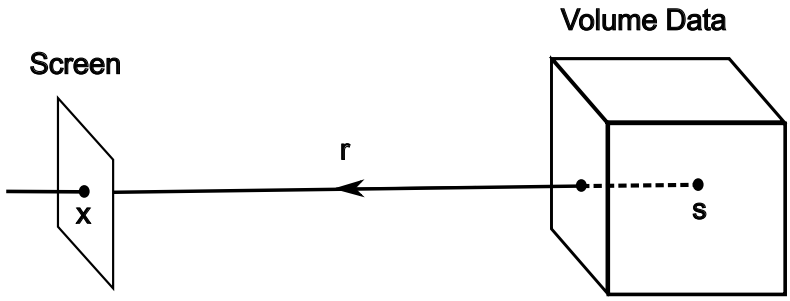


Figure 3.1: The model for direct volume rendering. At each point s, light is emitted. The ray **r** travels to the viewer's screen and contributes to the screen image at the coordinate **x**.

$$I_\lambda(\mathbf{x},\mathbf{r}) = \int_0^L C_\lambda(s)\mu(s)\exp\left(-\int_0^s \mu(t)dt\right)ds \tag{3.1}$$

where L denotes the length of ray **r**, or the maximum distance from the observer's eye. The integral sums up $C_\lambda(s)$, the energy emitted at each location s on ray **r**, with two weighting terms. If the density at point s is higher, the color emitted or reflected at point s needs to be brighter. To reflect this, $\mu(s)$ is used as a weight to $C_\lambda(s)$. The last part of integrand represents an extinction effect; the light is either attenuated or absorbed by particles between point s and the observer's eye.

Therefore, the goal of volume rendering algorithms is to evaluate Equation 3.1 efficiently and correctly. Equation 3.1 does not have a closed-form solution, and one has to compute $I_\lambda(\mathbf{x},\mathbf{r})$ numerically. Furthermore, the data is stored in a 3D structured grid, which is a regular sample from a continuous field.

3.1.1 Rendering Algorithms

3D Texture-based Algorithm

Direct volume rendering based on 3D texture mapping extensively utilizes the modern GPU architecture to accelerate the computation needed to evaluate Equation 3.1.

The integral in Equation 3.1 cannot be analytically evaluated as the data set is discretely sampled from slices. Therefore, the integration is approximated by the following discretized summation:

$$I_\lambda(\mathbf{x},\mathbf{r}) \approx \sum_{i=0}^{L/\Delta s} C_\lambda(s_i)\mu(s_i)\Delta s \prod_{j=0}^{i-1} e^{(-\mu(s_j)\Delta s)}$$

Moreover, the data itself is often not color information, but rather, it is values defined from a continuous scalar function $f(s)$. Considering the data value at s in a volume is represented by $f(s)$, the formula can be rewritten as follows:

$$\begin{aligned} I_\lambda(\mathbf{x},\mathbf{r}) &\approx \sum_{i=0}^{L/\Delta s} C_\lambda(f(s_i))\mu(f(s_i))\Delta s \prod_{j=0}^{i-1} e^{-\mu(f(s_j))\Delta s} \\ &\approx \sum_{i=0}^{L/\Delta s} C_\lambda(f(s_i))\alpha(s_i) \prod_{j=0}^{i-1}(1-\alpha(s_j)) \end{aligned} \tag{3.2}$$

where $\alpha(s_i)$, or α_i, replaces $\mu(f(s_i))\Delta s$, which we call *opacity*, and the exponential function is approximated with a first order Taylor series expansion. For simplicity, we also rename $C_\lambda(f(s_i))$ as $c(s_i)$, or C_i. In order to compute the values, we first sample necessary information, color and opacity, along the ray $\mathbf{r}$, and this is done by texture slicing.

Texture Slicing The volume rendering algorithms using 3D textures first slice the volume into many parallel 2D planes, called *slices*, to get samples along the ray. There are three major slicing methods, 1) object-aligned slicing, 2) viewport-aligned slicing, and 3) spherical shell slicing, and each of them lead to different rendering algorithms. Figure 3.2 illustrates the first two methods.

Object-aligned slicing slices a volume parallel to the coordinate axes of the volume. However, this approach became obsolete mainly for two reasons: First, the main drawback of this approach is that once the volume is rotated 90 degrees, the effect of artifacts caused by the empty space between slices becomes notable. A solution to the artifacts is using three sets of slices, each of which is parallel to the X, Y, and Z axes [RSEB$^+$00]. Based on the observer's viewing location, the best slicing method among the three is picked. However, this approach still has artifacts when the volume is rotated 45 degrees, and the burden of storing three copies of data is not negligible, especially on the scarce texture memory resource. Second, modern graphic processors

and libraries support 3D texture memory, which simplifies the slicing operations. With the 3D texture memory, the volume is sliced into planes perpendicular to the observer's perspective (viewport-aligned slicing). Whenever the observer rotates the volume, the slicing has to be re-computed. The values on planes (i.e. *fragments*) may not correspond to values on texture data (i.e. *voxels*), and thus, trilinear interpolations are needed to get the fragment values from the voxel information. Modern graphic processors can interactively perform trilinear interpolations in massively parallelized hardware.

The final slicing method is spherical shells [LHJ99], that is, concentric spheres centered on the viewer's point. One impractical aspect of this method is that each spherical shell needs to be tesselated into small polygons, generating a larger amount of polygons than by object-aligned slicing and viewport-aligned slicing. Therefore, slicing methods used in recent literature have converged to the viewport-aligned slicing method.

Compositing: Frame Buffer Blending The slices generated by the slicing are the sampling points for Equation 3.2. The intersection between the slice and a ray is a sampling point. This is illustrated in Figure 3.3. The dotted lines are the slices perpendicular to the viewing ray. Then the remaining problem is how to evaluate Equation 3.2. By reorganizing the equation into a recursive form as follows, we can accumulate the color and opacity:

$$C'_i = \alpha_i C_i + (1 - \alpha_i) C'_{i+1} \tag{3.3}$$

In this equation, C_i is the color value and α_i is the opacity on the current slice. The

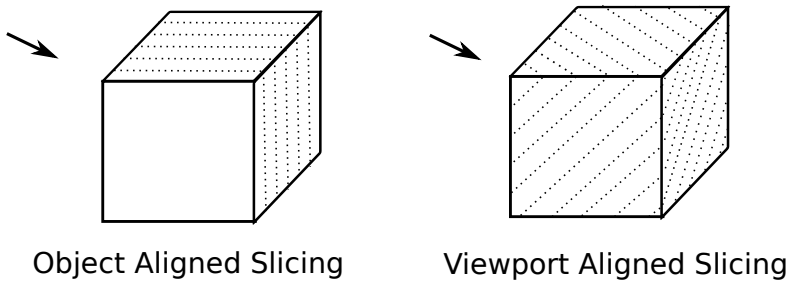


Figure 3.2: Slicing methods: object-aligned slicing versus viewport-aligned slicing. The viewport aligned slicing method changes the slicing depending on the location of the camera.

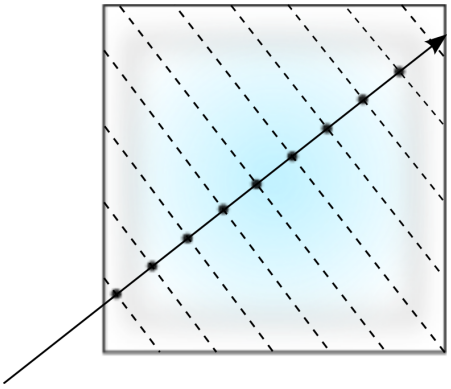


Figure 3.3: Sampling on the intersection of a ray and each slice. Because the samples are not aligned with the voxel positions, the sample points are interpolated from the neighborhood voxels.

term $\alpha_i C_i$ confirms that the amount of energy emitted is proportional to the density of particles, i.e., opacity. The extinction is explained by the second term. C'_{i+1} is the color intensity accumulated upto $(i+1)$-th slices from the farthest slice. The accumulated intensity contributes to the current intesity C'_i only with the weight $(1-\alpha_i)$ because of the extinction.

This formula requires blending the color information from the back-most slice to the front-most slice, called back-to-front compositing. Therefore, with the back-to-front compositing, the accumulated color in the i-th slice, C'_i, is blended with the color in the i-th slice.

From the point of frame buffer blending, for each fragment pixel p, C'_{i+1} is the color value accumulated up to $(i+1)$th slice, and C_i is the value read from the current texture slice. Then, the newly computed value for p up to ith slice is computed by Equation 3.3.

Ray Casting Algorithm

Ray casting [Lev88] computes the value of each pixel by shooting a ray into each pixel of the image plane. A ray is sampled from evenly spaced locations, and the value at each location is computed by trilinear interpolations among voxel data. The color and opacity values can be computed on either voxel points or sampled points on each ray. Because the ray tracing algorithms traverse pixel-by-pixel for each rendered frame, the

quality of image is shown to be superior to other approaches. However, without any support from GPU, the cost of computation is usually high.

To overcome the performance issue, many optimization techniques have been proposed [Lev90, YS93, DH92, MJC02, GBKG04]. Early ray termination by opacity thresholding is a technique to stop ray marching if the opacity accumulated has reached a threshold. If the opacity is over the threshold at a point on the ray, the new information behind the point does not contribute to the final image. Adaptive ray marching and empty space skipping are techniques that accelerate the ray marching process. Instead of marching rays in a fixed distance, these algorithms changes the distance dynamically. If the ray is passing through an empty space, the ray skips the region until it hits an object. The spatial information about empty space is utilized to determine whether or not the area that the ray is currently passing through is important. The information is organized as a hiearchical data structure, such as octree [Gar82] or k-d tree [SF90].

Recent advancement in programmable graphics hardware accelerates the ray casting algorithm on GPU. Kruger and Westermann [KW03] integrated early ray termination and empty-space skipping method into 3D texture based volume rendering to reduce per-fragment operations. Other approaches proposed similar ways of incorporating ray casting rendering logic into programmable GPUs [PBMH05, Sch05].

Nvidia's CUDA further extends the programmability of GPU stream processors and this allows programmers to have a high level of flexibility over the control of the stream processors. Since the CUDA library was released, several algorithms have been proposed [Kim08, KGB$^+$09, MRH10]. The common theme of GPU-based or CUDA-based algorithms is to modify the original ray casting algorithm to maximally exploit the hardware architecture. Challenges are how to fit the expensive computation into GPU so that the advanced hardware architecture can execute the computation quickly.

Kim [Kim08] proposed a streaming model to move the data between CPU and GPU. The rendering process consists of two stages: work list generation on CPU and rendering stage on GPU. The first stage is done on CPU because it requires travesal on a hierarchical data structure. The main performance benefit comes from the computation overlap between the data structure traversal and the rendering step, as well as from the highly parallel architecture on GPU. Kainz et al. [KGB$^+$09] presented a GPU-based

rendering algorithm for multiple volume data sets. They design the rendering step with two separate kernels, triangle kernel and pixel kernel. The triangle kernel transforms geometry, assigns triangles to viewport tiles, and computes coverage masks. The pixel kernels take care of per fragment processing, such as classification, opacity and color accumulation, and shading. Mensmann et al. [MRH10] introduced a slab-based raycasting algorithm on GPU. Slabs are aligned with image space whereas bricking is organized as an object order. A slab groups a set of rays and stores the data access pattern. Once the access pattern is computed, the information can be reused for the next slab. To expedite the memory access to the information, the slab is stored in shared memory. The shared memory in CUDA provides high bandwidth, and the algorithm benefits from multiple accesses to the shared memory.

3.2 Volume Rendering for Multi-Channel Data

All the rendering algorithms discussed so far are for general volume data sets. Those data sets have only one value defined at each voxel. In order to render multi-channel data, several approaches have been proposed. In addition, there are several commercial software packages available, as well as a software system from academia. In this section, we review both the algorithms and systems for multi-channel data.

3.2.1 Rendering Algorithms

The key problem in rendering algorithms for multi-channel data is how to blend multiple values to get the final image. Cai and Sakas [CS99] approached this problem by building a theoretical model for multi channel data. Later on, the issue moved on to how to efficiently blend multiple values to accelerate the rendering process [SR04, Lia07, LSW+07].

Cai and Sakas [CS99] presented three data intermixing schemes for multi channel data: 1) image level intensity intermixing, 2) accumulation level opacity intermixing, and 3) illumination model level parameter intermixing. Image level intensity intermixing renders multiple volumes separately on a single volume renderer, and the pixels from multiple rendered images are mixed together to get the final image. This approach

is simple, but it fails to provide depth information at the final image because the depth information is not provided on the image from each volume. The multi-channel algorithms correspond to this scheme. The second scheme accumulates the mixture of multiple opacity and color intensity information as one ray travels through voxel values. The final scheme, illumination model level intermixing, computes the opacity and intensity at each sampling point at a very early stage and uses the computed parameters in the rest of a rendering pipeline.

Cai and Sakas tried to propose a new illumination model for their last scheme, illumination model level parameter intermixing. In this model, however, they only considered two specific volumes: one volume attenuating light, e.g. X-ray data, and the other only emitting light without attenuation, e.g. the radiation distribution calculated by a physicist during Radiotherapy Treatment Planning. By restricting volumes to have either emission or attenuation, they could build a unified illumination model. This assumption, however, is apparently too limited, and there are few volume data sets satisfying this condition. The authors claimed that the final scheme is promising in that the image quality is better than the other two schemes, and the rendering speed is faster because once the parameters are computed, the rest of the rendering process is the same as a single volume rendering. They also argue that the accumulation level scheme is relatively slow because the opacity, and that the intensity values have to be computed on-the-fly. Moreover, defining transfer function is a very important process. When multiple volumes are intermixed, it is hard to assign a single color or opacity value. Cai and Sakas also mentioned that the transfer function problem is critical because without a good transfer function it is problematic to balance the visual contributions from multiple volumes.

Razdan et al. [RPFC01] proposed a way to render multicolor laser confocal microscope data. Confocal microscope images usually measure lights emitted from various fluorescent molecules, generating multiple channels. Their algorithm renders each channel's data separately and produces the final image by compositing. The optical model described in Equation 3.1 is used here, too. The intensity of the final composited image is obtained by summing up all the intensity values from each channel with its own weight values. For instance, if there are two channel data sets, say c_1 and c_2, pixel

values from c_1 and c_2 are added together with appropriate weight although they did not specifically mentioned how to define the weight values. The approach is simple and straightforward. The number of channels used here was three.

Later on, Schulze and Rice [SR04] succeeded in rendering four channel data sets. This was possible because modern GPUs store data in four fields: red, green, blue, and alpha. The first three channels are assigned to each color chanel, R, G, and B. The hue of the last channel is copied onto the alpha field. After rendering with only the first three channels into an intermediate image, the color of the last channel is superimposed on the image, leading to the final image. They also provided a user interface to adjust the hue value for the fourth channel to find the best color, i.e. the color that overlaps least with the color distributions of the first three channels. Unfortunately, this approach is impossible to be extended beyond four channels.

Liang et al.[Lia07] and [LSW+07] aimed to render data sets that have more than four channels. The classified values from each channel are integrated into one final value in a shader programs. They used weights to balance the multiple channels. They also provided a set of interfaces to adjust various optical properties, such as color, weight, or opacity for each channel.

3.2.2 Rendering Systems

Based on the theoretical studies presented in the previous section, there have been attempts to build a system specialized for multi-channel data. There are three well-known commercial software systems that neuroscientists use daily for their research. Amira [Vis11] provides a wide variety of data type, e.g., multi-channel volume, 3D mesh, vector field, etc., and supports many kinds of visual processing, e.g., image segmentation, image filtering, and surface editing. The control panel that allows users to connect available modules to each other simplifies the process of handling the same type of workflows many times. Amira can load multiple data, each of which represents one channel, and can combine them as one volume.

Imaris [BIT11] is designed for 3D microscopy data sets and provides data visualization, analysis, and segmentation. For multi-channel data, users can adjust the weight of each channel in order to manipulate the transfer function. Imaris, like Amira,

also supports large-scale images that do not fit in the main memory. Volocity [Per11] also fully supports for multi-channel microscopy data sets. One can load multi-channel data, adjust weights for each channel, and render with all available channels.

Finally, Wan et al. [WOCH09] proposed another system for multi-channel confocal microscopy data. It aims to deliver a system best suited for neurobiology research by supporting interactive volume rendering with multi-dimensional transfer functions and multi-views. Users can adjust the rendering parameters interactively to acquire better image quality. Wan et al. adopted the image level intermixing introduced by Cai and Sakas [CS99] to blend multiple channels.

All the systems introduced here can successfully render multi-channel data. One can navigate the volume, pan in and out, and rotate, etc, which is all basic navigation. Large volume support, that is, the ability to handle data sets larger than the size of system memory, is also one of default functionality. Despite the basic functionality, there is a significant gap between the rendering systems for multi-channel data and for single-channel data. More specifically, the support for transfer function is significantly lacking compared to what the rendering systems for single-channel data provide as default. We will discuss this issue in detail in the next chapter.

3.3 Summary

We have presented the basic building blocks for volume rendering. The images generated by volume rendering are evaluations of the rendering equation integral. Because the integral cannot be computed analytically, the value is numerically computed. There are two main algorithms for the computation: 3D texture-based and ray casting algrotihm. For multi-channel data, several algorithms have been proposed to accurately render the data. Several commercial software systems support multi-channel volume rendering.

Chapter 4

Transfer Function

Volume rendering has three major pipelines: data acquisition, volume rendering, and data exploration via transfer function manipulation. We have discussed in details about the first (Chapter 2) and second (Chapter 3) steps. This chapter presents the remaining part, transfer function exploration. We first define transfer function in volume rendering and the role in visual exploration. We then introduce previous developments in transfer function, in particular multi-dimensional transfer function. Those developments have some limitation when dealing with multi-channel data. This limitation raises a research problem in order to explore multi-channel data effectively.

4.1 Definition of Transfer Function

As explained in Chapter 2, the volume data has one or multiple values at each voxel. Volume rendering model the values as the density of particles at the voxel. Transfer function is a function that maps the density value (called "intensity") to opacity and color information. Namely, transfer function tf for single channel is defined as:

$$tf : i \in \mathbb{R} \rightarrow (R, G, B, \alpha) \in [0,1]^4 \tag{4.1}$$

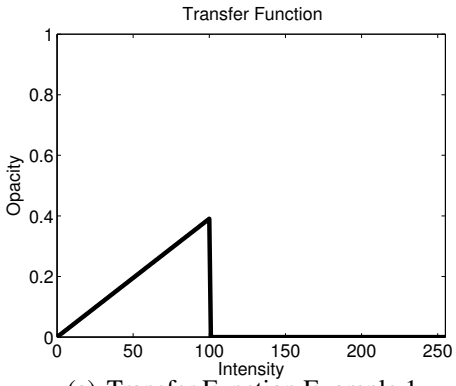
where i denotes the intensity of a voxel. This form is the simplest among many possible ways of defining a transfer function. For example, the domain can be multi-dimensional, instead of one-dimensional which is constructed by intensity. We refer to this form of

transfer function as *one-dimensional (1D) transfer function*. This form is used in many volume rendering systems as a default transfer function due to its simplicity. A few examples of 1D transfer function can be 1) a linear function that maps a white color to the highest intensity and a black color to the lowest intensity, 2) a window function that maps a color only to the intensity of a certain range either as a step function or a triangular function, or 3) the superposition of multiple step and/or triangular functions. By adjusting the width of step or triangular functions one can eliminates parts that are not interesting. For multi-channel data, the domain is a subspace of C-dimensional space:

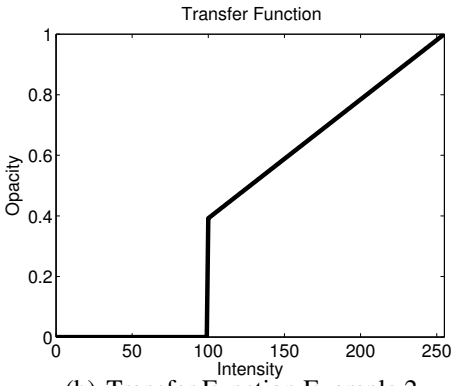
$$tf : \vec{i} \in \mathbb{R}^C \rightarrow (R, G, B, \alpha) \in [0,1]^4 \tag{4.2}$$

The goal of searching for a good transfer function is to classify or emphasize areas of interest, such as a range of intensity values or an object in the data set, with distinct colors and opacities. The areas of interest changes depending on users' interest. For example, if one is interested in a low density region and would like to see what low intensity regions look like, he or she needs a way to highlight the low intensity values. Figure 4.1 shows an example of how two different settings of transfer function can highlight different parts of the data. Figure 4.1 shows the effect of 1D transfer function. Figure 4.1(a) shows a transfer function emphasizing low intensity points. In this function, only the low intensity values are mapped to non-zero values, whereas intensities greater than 100 are mapped to zero. With this transfer function, the resulting image of volume rendering is shown in Figure 4.1(c). As shown, low intensity values render the soft skin of a human head. On the other hand, the second example of transfer function shown in Figure 4.1(b) emphasizes high intensity values. The opacity for intensity of 100 or greater linearly increases while intensity less than 100 is mapped to zero. This transfer function changes the original volume to show only the bones inside the soft skins as shown in Figure 4.1(d).

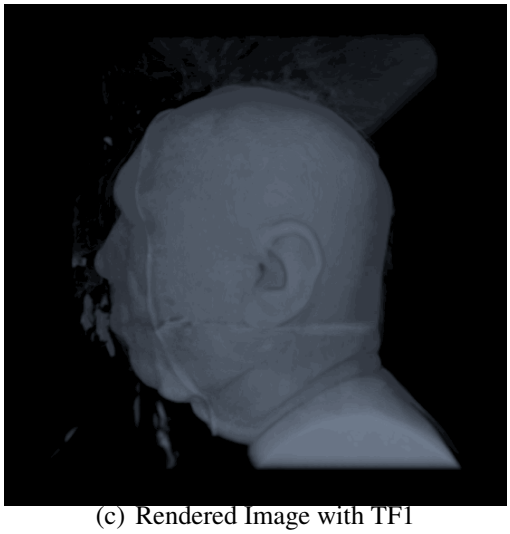
The RGB colors of voxels can be determined in the same way. Three separate functions for the red, green, and blue colors set the color intensity. For example, one can color the bones white and the soft skin the color of human skin. Therefore, transfer function is a useful tool to extract a part of the entire volume data or to assign realistic



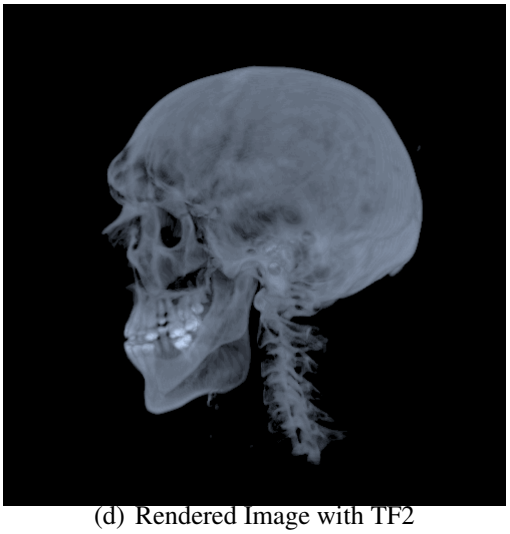
(a) Transfer Function Example 1



(b) Transfer Function Example 2



(c) Rendered Image with TF1



(d) Rendered Image with TF2

Figure 4.1: Transfer function examples and volume rendered images after applying the trasnfer function to the data. The data is a CT scan from the Visible Human Project by the National Library of Medicine of National Institutes of Health, USA. The data size is 256 x 128 x 256. The first data emphasize regions with low intensity, which is the soft skin of the face. On the other hand, the second transfer function is set to emphasize high intensity values. For this goal, the transfer function cuts off low intensity data and maps it to zero. With this transfer function, one can see bones inside the soft skin.

or artificial color so that viewers can easily understand each part of the volume.

However, one of the challenges in transfer function design is that the great amount of flexibility often makes users difficult to define a good transfer function. Users have to explore many — or infinitely many — possible ways to find a good transfer function, which is often very time consuming. Therefore, transfer functions have been an active area of research in the field of volume rendering [Kin02, PLS+00], due to the significance and the difficulty of finding a good transfer function in direct volume rendering.

4.2 Multi-Dimensional Transfer Function

4.2.1 Definition

The limitations of 1D transfer function are that it can only distinguish different data values from one another, regardless of their neighborhood, and that it does not translate to multi-channel data sets, where every voxel consists of multiple data values (such as for multiple fluorescent proteins in confocal microscopy). These problems prevent users from assigning colors differently for the areas of interest. For example, if an organ in MRI data has the same intensity value with soft skins, it is impossible to separate the organ from the softskin by different colors.

In order to handle this problem, Levoy [Lev88] and Kindlmann and Durkin [KD98] proposed using derivatives of the intensity values, such as the magnitude of first and/or second-order gradient. If the first-order gradient is large at a voxel, it means that the intensity changes rapidly around the voxel. This type of intensity change usually occurs around boundaries of an object. Therefore, by taking additional values into account, two voxels of the same intensity can be separated.

They improve 1D transfer function for single channel data by adding additional features derived from the original intensity value. Namely, the transfer function extends the intensity domain to multi-dimensional space:

$$tf : (i,g) \in \mathbb{R}^2 \rightarrow (R,G,B,\alpha) \in [0,1]^4 \tag{4.3}$$

where i is intensity as in 1D transfer function, and g denotes the gradient magnitude:

$$g = \left(\frac{\partial I}{\partial x}^2 + \frac{\partial I}{\partial y}^2 + \frac{\partial I}{\partial z}^2 \right)^{1/2} \tag{4.4}$$

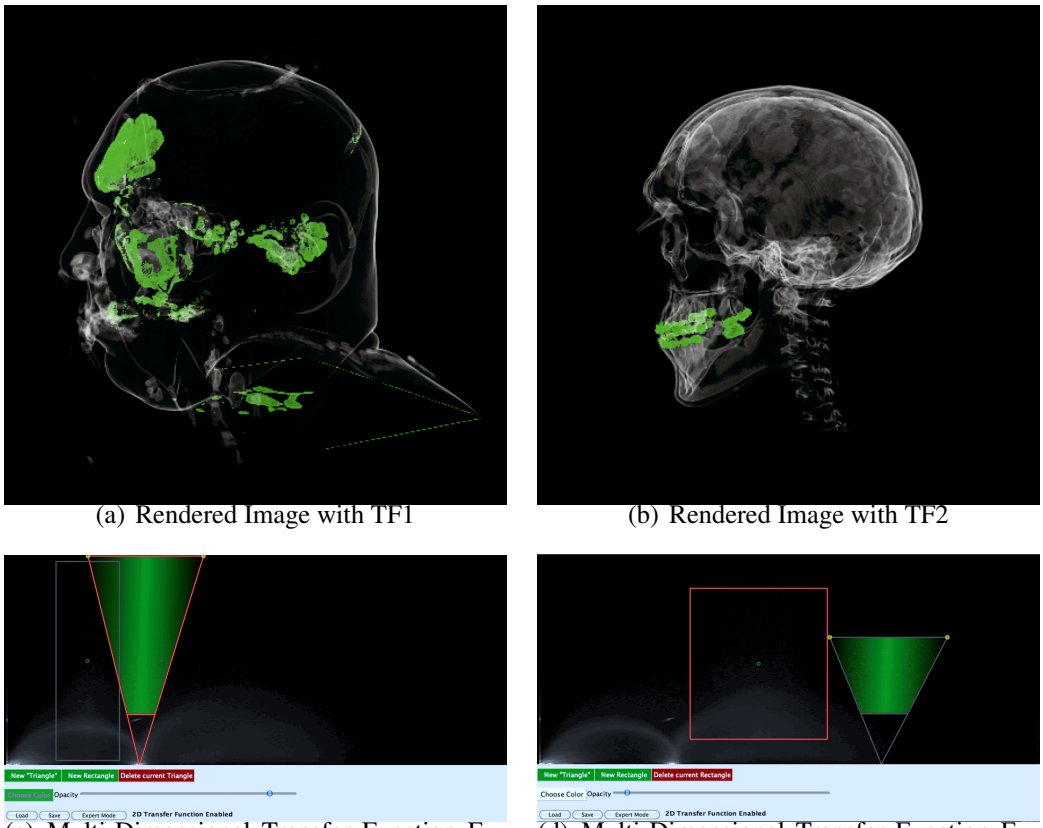
Therefore, a voxel is described with two properties, its original intensity and gradient magnitude at the position. Two points that have the same intensity but different gradient magnitude now can have different colors and opacity. This is impossible with 1D transfer function. Because the domain is multi-dimensional, this type of transfer function is called *multi-dimensional* transfer function.

Kniss et al. [KKH02] further strenghtened the idea by proposing dual domain interaction with widgets. In this work, users have two screens, one for rendered image and the other for transfer function domain. Figure 4.2 is an example of dual domain. Whenever users update the multi-dimensional transfer function, the users can see how the update changes the final image. Additional widgets and tools are proposed to effectively navigate the transfer function domain.

Figure 4.2 shows an example of multi-dimensional transfer function. The images are captured from an open-source volume rendering software, ImageVis3D [SCI], from the Scientific Computing and Imaging Institute at the University of Utah. This example shows the same data set used in Figure 4.1. However, the multi-dimensional transfer function shows that it is much more expressive than the 1D transfer function. In Figure 4.2(a), the boundary of the soft skin is colored in gray, while the inner organs, e.g. sinuses and inner ear, are highlighted in green. The domain of this data set and its associated color assignment are shown in Figure 4.2(c). Figure 4.2(b) shows another example. In this example, the boundaries of the skull are rendered in gray. The teeth are captured and colored in green. As shown in these figures, gradient magnitude helps users identify boundaries between materials and multi-dimensional domain can capture areas of interest more effectively.

4.2.2 Different Methods for Effective Classification

Besides the graident magnitude of intensity, additional information to construct the transfer function domain was proposed [KWTM03, HKG00, RBS05, CM08, CM09,



(a) Rendered Image with TF1

(b) Rendered Image with TF2

(c) Multi-Dimensional Transfer Function Example 1

(d) Multi-Dimensional Transfer Function Example 2

Figure 4.2: Multi-dimensional transfer function examples and volume rendered images after applying the transfer function to the data. The data is a CT scan from the Visible Human Project by the National Library of Medicine of National Institutes of Health, USA. The data size is 256 x 128 x 256. The first data set emphasizes the boundary of soft skins as well as inner organs, such as sinuses and inner ear. Two different colors, gray and green, are used to highlight areas of interest. In the second example, shown in Figure 4.2(b), the boundaries of skull and teeth are highlighted. One can clearly see the teeth colored in green as well as the overall structure of the skull. These examples show that multi-dimensional transfer function is much more expressive than 1D transfer function. For this rendering, we used an open-source volume rendering software, ImageVis3D [SCI], from the Scientific Computing and Imaging Institute at the University of Utah.

Table 4.1: Various approaches for multi-dimensional transfer function. Each approach proposes different information for the sake of better classification. Additional information is shown to isolate the areas of interest more effectively.

Approach	Information
Levoy [Lev88] Kindlmann and Durkin [KD98] Kniss et al. [KKH02, KKH01]	Gradient
Hladuvka et al. [HKG00] Kindlmann et al. [KWTM03]	Curvature
Röttger et al. [RBS05]	Spatial information
Correa and Ma [CM08]	Size of features
Correa and Ma [CM09]	Occlusion
Lindholm et al. [LLLa$^+$10]	Local material distribution

LLLa$^+$10]. Table 4.1 summarizes previous approaches in defining a new domain for multi-dimensional transfer function. Although the approaches listed in Table 4.1 utilize different information, they share the same goals: finding areas of interest effectively and assigning different colors and opacity to those areas. The essence of these ideas is that the additional information from the data points can classify the features of interest, otherwise impossible with only intensity and gradient magnitude.

Curvature information [KWTM03, HKG00] was shown to be useful in feature localization. Hladuvka et al. [HKG00] defines the transfer function domain with principal curvature magnitudes. The two principal curvatures can capture 1) plane, when $\kappa_1 = \kappa_2 = 0$, 2) parabolic cylinder when $\kappa_1 > \kappa_2 = 0$ or $\kappa_1 = 0 > \kappa_2$, 3) paraboloid when $\kappa_1 \kappa_2 > 0$, and 4) hyperbolic paraboloid when $\kappa_1 \kappa_2 < 0$. This helps users to identify surface structures, thickness, and smoothness of layers in the volume data. Kindlmann et al. [KWTM03] utilize the curvature information in the multi-dimensional transfer function domain. It is also shown that the curvature-based transfer function improves non-photorealistic volume rendering by emphasizing ridges and valleys. They also used principal curvatures κ_1 and κ_2 as the main axes for their transfer function domain.

While the first and second derivative information is to capture the shape of the objects in the data and it was shown to be successful, it lacks spatial information. That is, it is not possible to distinguish a point in the organ with another point on the soft skin. This is because intensity and graident does not contain where the two points are located in the original data. Therefore, researchers started to add spatial information to cluster points that are located closely to each other in the spatial domain.

Röttger et al. [RBS05] incorporate spatial information into multi-dimensional transfer functions. Their algorithm automatically classifies the transfer function domain into many coherent groups, each of which represents a spatial feature in the rendered data. The algorithm first creates a two-dimensional transfer function domain with intensity and gradient magnitude. They assume that the points mapped to the same coordinate in the domain are likely to belong to the same feature if the points are close to each other spatially.

Correa et al. [CM08] considered relative sizes of features as an alternative domain. The size of features is encoded with scale fields. The scale field maps a voxel x to the local scale of the feature containing x. Their size-based transfer function which is derived from the scale field can distinguish large and small features that have similar or identical intensity. This is especially useful in angiography to discern aneurysms from normal vessels. Another example shown in the paper is CT scan data showing bones and vessels. Because the sizes for the voxel inside bones and vessels are different, the transfer function allows users to assign different color and opacity to bones and vessels.

Correa and Ma [CM09] utilize the ambient occlusion information to classify features. This information is shown to be useful in detecting features inside the volume data, which is often highly occluded. For example, MRI data has multiple layers, such as venticular anatomy, skull, brain, and soft skin. The innermost objects have a large degree of occlusion, whereas the softskin has minimal occlusion compared to other features. Another advantage of using occlusion for transfer function design is that the occlusion information also encodes the spatial coherence, which is also proved to be important in other approaches [RBS05, CM08].

Lindholm et al. [LLLa$^+$10] adds material information to extract the spatial information. Their observation is that the voxels close to each other (i.e., spatial information)

have similar material characteristics (i.e., material information). The transfer function design is done as usual at first, but it is further conditioned with the extra material information. The material information is defined either by explicit user interaction or by automatic peak detection.

4.2.3 Transfer Function Domain Exploration

Another challenge in transfer function design is how to define color and opacity on the transfer function domain. Multi-dimensional transfer function is represented by multi-dimensional histogram. Kniss et al. [KKH02, KKH01] proposed a set of tools that can effectively define colors and opacity on the 2D transfer function domain. They proposed two types of widgets, the shapes of which are either triangular or rectangular. Users can overlay widgets on the transfer function domain to notate areas of interest. A color and opactiy pattern is defined for each widget as shown in Figure 4.2.

Although the manual way of defining a transfer function with widgets is popular and intuitive, Tzeng et al. [TLM03] further proposed a different interface. The system defines color and opacity as usual, but, after that, users can paint on the volume slice to tell the system where the areas of interest are. The information given by users is fed into a classification algorithm based on neural network, filtering only the similar materials to what users selected. Huang and Ma [HM03] tackled the same problem with a different method. Their algorithm finds coherent structures by using a region growing algorithm. The algorithm starts with a set of seed points, and neighbor points are added into the set if the points are similar to the seed points in terms of original intensity and gradient magnitude. Finally, an algorithm based on non-parametric clustering on the transfer function domain was proposed [MWCE09]. The clusters provides starting points for feature identification and volume exploration. All these approaches allows users to easily explore the transfer function domain and classify the features of interest with minimum effort.

4.3 Transfer Function for Multi-Channel Data

As discussed in the previous section, multi dimensional transfer function is widely used in volume exploration. In this section, we turn to the multi-channel data. Our claim is that there is a significant gap between the transfer function capability for single-channel data and for multi-channel data.

The rendering systems for multi-channel data have not adopted the advanced technologies developed for single-channel data. The most common way to change the appearance of multi-channel data is by manipulating each channel. Each channel is weighted differently so that one channel stands out visibly. Scientists usually turn on and off each channel to investigate them one by one, or to make one channel stand out from other channels. The parameters for defining a transfer function include offset, low and high thresholds, and/or multi-dimensional domain for each channel [WOCH09].

With the advanced transfer functions, it is possible to highlight: 1) only the regions where channel one and two are bright, 2) the regions where two channels have high intensity around cell boundaries, or 3) the regions that have high curvature values. These are just a few examples that can be provided by the technologies developed for the single-channel data.

The biggest challenge in providing such advanced transfer functions for multi-channel data is that human perception of four or higher dimensional data is very limited. However, suppose that we have only three channels, and that we would like to employ gradient magnitude values to locate boundaries of objects in the data. Then the gradient magnitude values for each channel occupy three-dimensions, and with the intensity domain, 6 dimensional space is required. In order to find features of interest, it is required to handle the entire domain all at once. The entire domain can reveal a set of points that have a similar properties. As illustrated in Figure 4.3, there exists a huge gap between transfer function technologies for single-channel data and for multi-channel data. In order to narrow the gap between the two, it is critical to keep the dimensionality of the domain for multi-channel data to 2D or 3D. Otherwise, it is impossible to “see” the domain, or one has to compromise and break down the domain into smaller subspaces, which misses the entire picture of the original domain.

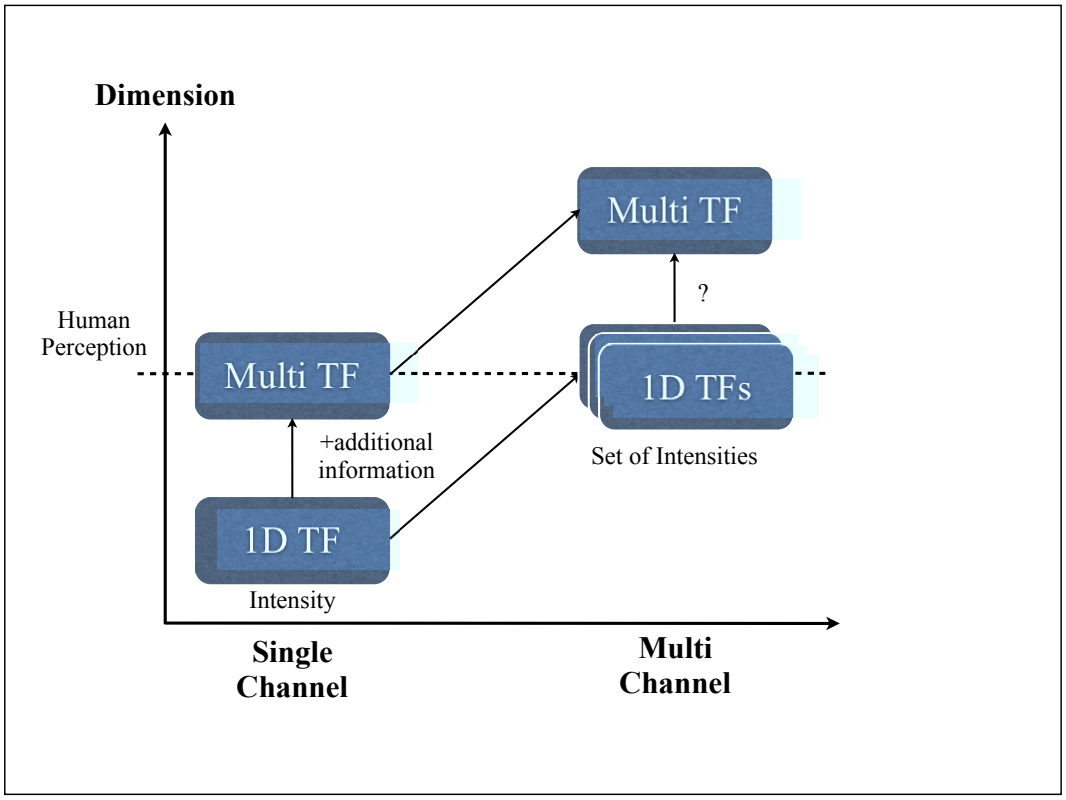


Figure 4.3: The challenge in constructing multi-dimensional transfer function for multi-channel data. As the human perception is limited to three dimensional space, constructing multi-dimensional transfer function is challenging.

4.4 Summary

We have defined 1D and multi-dimensional transfer function and have shown their importance in volume rendering. Advanced transfer function exploration enables viewers to successfully find areas of interest and visually highlight those areas. Numerous approaches have shown that different features of intensity data can isolate areas of interest by constructing multi-dimensional domain with their own features. The current transfer function support for multi-channel data is very limited, mainly due to the limitation in the maximum dimensionality. It is very difficult to navigate four or higher dimensional space. However, having multi-dimensional transfer function for multi-channel data requires extra features for each channel, which can easily make the domain above 3D space.

Acknowledgement

This chapter, in part, has been published as "Dimensionality reduction on multi-dimensional transfer functions for multi-channel volume data sets" by Han Suk Kim, Jürgen P. Schulze, Angela C. Cone, Gina E. Sosinsky, Maryann E. Martone in Information Visualization Journal, Palgrave Macmillan in 2010 [KSC+10a]. The dissertation author was the primary investigator and author of this paper.

This chapter, in part, has been published as "Multichannel transfer function with dimensionality reduction" by Han Suk Kim, Jürgen P. Schulze, Angela C. Cone, Gina E. Sosinsky, Maryann E. Martone, in the proceedings of SPIE Visualization and Data Analysis conference in 2010 [KSC+10b]. The dissertation author was the primary investigator and author of this paper.

Chapter 5

Multi-Dimensional Transfer Function for Multi-Channel Data

5.1 Introduction

In volume rendering, the transfer function maps visual properties, such as color and opacity (alpha), to each voxel in a volumetric data set (i.e., data filling a three dimensional space). A carefully designed transfer function can reveal much more visual information than a simple one, and thus is very valuable for scientists exploring 3D volume data sets [PLS+00]. Effective transfer functions will identify the different materials, or features of a data set, such as bone or skin in medical data sets, or cell nuclei or protein structures in microscopical images [Kin02]. It is desirable that the transfer function design process be done in real time, to minimize the duration of the partially manual design process.

As discussed in Chapter 4, multi-dimensional transfer function requires multiple mathematical values and those values increase the dimensionality of the transfer function domain. Our goal is to provide a method to design a transfer function for multi-channel data while preserving the information that a multi-dimensional transfer function domain has. We propose a method that provides a lower dimensional representation of the original high dimensional domain so that users can design multi-dimensional transfer functions while keeping all the rich information in the high dimensional space.

Our method employs three different dimensionality reduction algorithms: Principal Component Analysis (PCA), Isomap, and Locally Linear Embedding (LLE). Although there have been many efforts in the machine learning community to propose better algorithms for dimensionality reduction algorithm, we decided to use Isomap and LLE because they are two representative algorithms among many others. By using two different "advanced" algorithms, we present the effectiveness of the two algorithms in a field they have not previously been applied to. We investigate our approach with microscopy data sets to test the following scenarios that we often encounter in analyzing multi-channel data with transfer function:

Multiple Features When we have little information about the data and when we do not have enough insight about which features can effectively isolate areas of interest, without our approach one needs to iteratively compare with 2 or 3 features. Then $O(n^2)$ many comparisons are needed where n is the total number of features. This takes considerably longer, and it is hard to capture the shape or distribution of the entire multi-dimensional domain. Because our approach keeps the shape of the multi-dimensional domain while it provides a 3D view of the domain, visualization scientists can easily understand how the voxels are distributed across the multi-dimensional domain. For this scenario, a data set is prepared in such a way that we use up to 24 features, which leads to a transfer function domain with up to 24. This is the domain we then run our dimensionality reduction algorithms. We find that the result from it helps understand the high-dimensional transfer function domain, which is shown by a more useful coloring of the volume data set.

Multiple Channels Assuming that we have more knowledge about the data and we know that some data channels are more important for the visualization than others. Although many features can possibly isolate more areas of interest, if we are certain about which features we would like to use, then the remaining features may be considered less important. The benefit of dimensionality reduction still holds here because multi-channel data has 3 or 4 intensities and adding one more feature increases the dimension to 6 or 8. The difference compared to the previous scenario is, however, that the original domain is much smaller, i.e. 6D in our

example. We demonstrate our results with a light microscopy data set of a blood vessel in a mouse's hippocampus region that was stained for specific proteins.

This chapter is organized as follows: Section 5.2 discusses previous approaches on dimensionality reduction algorithms. Then, Section 5.3 reviews the three dimensionality reduction algorithms we employed for our work. Section 5.4 presents how we apply dimensionality reduction algorithms to automate transfer function design for multi-channel data sets. Section 5.5 outlines the experiments validating our approach and Section 5.6 shows results from dimensionality reductions and volume rendering. Finally, Section 5.7 discusses how our algorithm fits in many existing algorithms and Section 5.8 concludes this chapter.

5.2 Related Work

Our new method combines algorithms from two different disciplines: multi-dimensional transfer functions from volume rendering and dimensionality reduction algorithms from machine learning. As we discussed transfer function in details in the previous chapter, in the following, we discuss previous work on dimensionality reduction algorithms related to our approach.

5.2.1 Dimensionality Reduction

Besides the dimensionality reduction algorithms discussed in Section 5.3, many alternative algorithms have been proposed. Although PCA has been successfully used for a long time in many fields, recent development in non-linear dimensionality reduction algorithms have shown great success in analyzing complex non-linear high-dimensional data [WS06, WSZS07, RS00, BN03, DG03a]. Each approach attempts to preserve different metrics and constructs different optimization problems under different constraints. However, the problem they all try to solve is the same, finding a low dimensional representation that faithfully describes original high-dimensional data.

In scientific visualization, self-organizing maps [KSH01] and K-means clustering [Mac67] have widely been used for data exploration. Although each algorithm has

shown strength in data analysis to display the distribution of high-dimensional data, the results are subject to a set of initial reference vectors or initial K-means that users initially assign. The non-linear dimensionality reduction algorithms, on the other hand, require a minimum number of free parameters, e.g. K in the K-nearest neighborhood algorithm, and they find a low-dimensional representation that preserves relative distances in high-dimensional space. Moreover, non-linear dimensionality reduction algorithms produce a single global optimal point, as opposed to many local minima in self-organizing map or in K-means clustering.

Lewis et al. [LHWS08] applied K-means clustering, PCA, Isomap, and Maximum Variance Unfolding [WS06, WSZS07] to oceanographic multivariate data in an approach similar to ours. They use clustering and classification of data into multiple disjoint groups, which is parallel to our work in that we focus on color encoding for transfer function design and feature discovery in volume visualization to emphasize areas of interest in biological images.

5.3 Dimensionality Reduction

In this section, we briefly review the three dimensionality reduction algorithms that we employ, namely, 1) Principal Component Analysis (PCA), 2) Isomap, and 3) Locally Linear Embedding. We discuss these algorithms in terms of what each of the algorithms preserves in the low-dimensional representation and what they minimize/maximize. Intuitively, the inputs to dimensionality reduction algorithms are a set of vectors in $D > 3$ dimensional space and the outputs are the same number of points but in 3D space. Preserved between the inputs and outputs are the distances between points.

Let $X_1, \ldots, X_N$ be inputs in high dimensional space, $\mathbb{R}^D$. The goal of dimensionality reduction algorithms is to find a low dimensional representation $Y_1, \ldots, Y_N \in \mathbb{R}^d$ such that $d \ll D$ and $\{Y_i\}_{i=1}^N$ faithfully explains the original data $\{X_i\}_{i=1}^N$.

5.3.1 Principal Component Analysis

The main idea of Principal Component Analysis (PCA) is to find a subspace that maximizes the variance when $\{X_i\}_{i=1}^N$ are projected to the subspace. For example, if X_i's are spread out on or near the x-axis in $\mathbb{R}^2$, a good 1D subspace would be the x-axis, rather than the y-axis: all the points projected to the y-axis are around $y = 0$ whereas projecting to the x-axis loses less information by preserving the variance in 2D. Therefore, PCA maximizes the variance of projected sample points to direction $\hat{u}$:

$$\hat{u} = \max_{\vec{u}} \frac{1}{N} \sum_{i=1}^{N} (\vec{X}_i \cdot \vec{u})^2$$

The solution for this optimization problem is computed by singular value decomposition (SVD) [DHS00]. SVD results in the eigenvector with the largest eigenvalue of covariance matrix C of sample points X_i. Namely, for a given set of points $\{\vec{X}_i\}_{i=1}^N$, PCA computes the covariance matrix C of X_i's and solves eigenvalue equations $C\vec{u} = \lambda\vec{u}$. Then the top d eigenvectors that yield the largest eigenvalues are the direction $\hat{u}_j$ ($j = 1 \ldots d$) which achieves maximum variance. The jth component of Y_i ($j = 1 \ldots d$) can be easily computed with the inner product of X_i and $\hat{u}_j$, i.e. $Y_{ij} = X_i \cdot \hat{u}_j$. More details about how to derive the solution can be found in Appendix A.

Due to the simplicity of the algorithm, albeit computationally expensive, PCA has been widely used in many fields. Note also that its solution is the global minimum. However, the limitation is the assumption that the sample points are on or near a linear subspace.

5.3.2 Isomap

Isomap [TdSL00, ST03] is an extension of multidimensional scaling (MDS) [BG97] for finding a low-dimensional representation of samples that have nonlinear shapes. The algorithm assumes that the observation points in high-dimensional space are a sample from an intrinsic lower dimensional manifold. The key idea here is that the geodesic distance between two points, X_i and X_j, not the Euclidean distance in $\mathbb{R}^D$, captures the shape of non-linear structures in high-dimensional space. One canonical example widely used is a Swiss roll shaped plane. A long strip shaped plane is rolled

into a spiral. Geodesic distance is the distance between two points on the Swiss roll when traveling only on the plane. Then, MDS constructs a low-dimensional representation $\{Y_i\}_{i=1}^{N}$ where the distance between two points in Y is closest to the geodesic distance.

The algorithm has three steps. The first step is to construct a neighborhood graph based on Euclidean distance in the high-dimensional space. For each point X_i, the distance to other points, $\|X_i - X_j\|_2$ is computed, and the closest K points are picked. The distance is used as the weight between the two nodes, X_i and X_j, in the neighborhood graph, but if two nodes are not close to each other, i.e., they are not in the K-nearest neighborhood to each other, no edge exists. The second step is to compute the shortest path between any two nodes. The resulting information, shortest distances between nodes, encodes the geodesic distance. The distance approximately measures the distance on the manifold. For example, if we have points sampled from a sphere, the geodesic distance represents the shortest distance when traveling only on the surface of sphere. Thus, this graph captures the nonlinear manifold. The last step is running MDS on the graph to find a low-dimensional representation preserving the pairwise geodesic distances.

The computational cost is higher than PCA because of the first and second step. A naive implementation of the K-nearest neighborhood can go up to $O(N^2D)$. However, a sophisticated implementation can optimize the process, and it runs much faster. Furthermore, the process is embarrassingly parallelizable, so OpenMP [DM98] can be utilized to boost the performance on a multi-core shared memory system. The second step, with Dijkstra's algorithm [CLRS01] and it's naive implementation, can run in $O(N^2)$, although it is known that a Fibonacci heap can reduce the asymptotic time down to $O(N \log N + E)$, where E is the number of edges in the graph.

The cost can be further reduced by using landmarks [ST03]. Instead of using all the points in the analysis, which are redundant in most cases, only a small set of randomly chosen points, called landmarks, are considered. This algorithm dramatically reduces the computational cost while producing almost the same quality of the resulting low-dimensional representation.

5.3.3 Locally Linear Embedding

Locally Linear Embedding (LLE) [RS00] is another popular nonlinear dimensionality reduction algorithm. Whereas Isomap tries to preserve geodesic distance in the manifold, LLE preserves locally linear structures. Namely, LLE exploits the relationship between a point and its neighborhood, which is represented as a linear patch around the point. The linear patches approximating the data are analogous to triangles approximating a sphere in computer graphics. Moreover, the locally linear structures are invariant to translation, scaling, and rotation. This means the algorithm has the freedom to take each patch from tangled objects in high dimensional space and maps into a low dimensional space.

The algorithm starts with the same step as in Isomap, computing the K-nearest neighborhood K. Then, it finds W_{ij} such that W_{ij} minimizes $|X_i - \sum_j W_{ij} X_j|^2$ with the constraint of 1) $W_{ij} = 0$ if X_j is not K-nearest neighborhood of X_i and 2) $\sum_j W_{ij} = 1$. Intuitively, X_i lies on the plane, the linear patch, constructed by X_j's, and X_i can be reconstructed by interpolating X_j's with weight W_{ij} with a minimum error. The last step is to find low-dimensional coordinates Y_i by minimizing the following embedding cost function:

$$\Phi(Y) = \sum_i |Y_i - \sum_j W_{ij} Y_j|^2 \tag{5.1}$$

where this optimization problem is a quadratic form for Y_i's and W_{ij}'s are the weights computed in the previous step. The solution for this problem is found by solving, again, an eigenvalue problem. For this problem, however, we need $d+1$ eigenvectors of matrix $(I-W)^T(I-W)$ that have smallest positive $d+1$ eigenvalues as opposed to largest eigenvalues as in PCA and Isomap.

The benefit of LLE is that: 1) the algorithm is straightforward and easy to understand; 2) it does not require expensive processing to construct an input for the eigenvalue problem, as opposed to shortest path algorithm in Isomap; and 3) the matrix constructed by step 2 is usually very sparse, which consumes less memory, whereas Isomap produces a dense matrix. However, the computational cost of finding eigenvectors of smallest eigenvalues is usually much higher than those of largest eigenvalues.

5.4 Transfer Function Design

In this section, we discuss that how the dimensionality reduction algorithms discussed in Section 5.3 are employed in transfer function design. We already discussed the limitation of current multi-dimensional transfer function in multi-channel data in Chapter 4. In order to overcome the limitation, we propose our solution that utilizes dimensionality reduction algorithms and explain in detail how our approach works.

Principal component analysis (PCA) has been widely used in the visualization community, whereas Isomap and LLE are relatively new algorithms that can reduce the dimensionality of data to a low dimensional representation in a non-linear fashion. All three algorithms reduce the dimensionality while preserving relative distances between points in high-dimensional space. Therefore, two points close to each other in the domain after applying dimensionality reduction algorithms are also close to each other in the original domain. However, we claim that non-linear dimensionality reduction algorithms are more appropriate for our problem, because Isomap and LLE find the low dimensional representation with a non-linear mapping and because the points in the transfer function domain are often distributed in a non-linear way.

The distance-preserving characteristics of dimensionality reduction algorithms are crucial in the transfer function domain, because the concept of transfer function design is based on the assumption that points close to each other in the data domain share the same or similar properties, and that if those points are assigned to the same or similar colors, we will perceive them as part of the same object. While the dimensionality reduction algorithms automatically generate a domain, the freedom of visualizing the resulting lower dimensional domain stays the same as in traditional multi-channel transfer function domains. One can examine the domain to find areas of interest using any type of widgets to map color and opacity to the data values. Another advantage of dimensionality reduction is that the number of features is less restricted in our proposed method and users are free to add additional fields to the transfer function domain. Once the domain has enough information to isolate areas of interest in the volume, the high dimensionality is reduced to the usual up to three dimensions.

Figure 5.1 illustrates how dimensionality reduction algorithms can overcome the limitation in multi-dimensional transfer function. In order to define a transfer function,

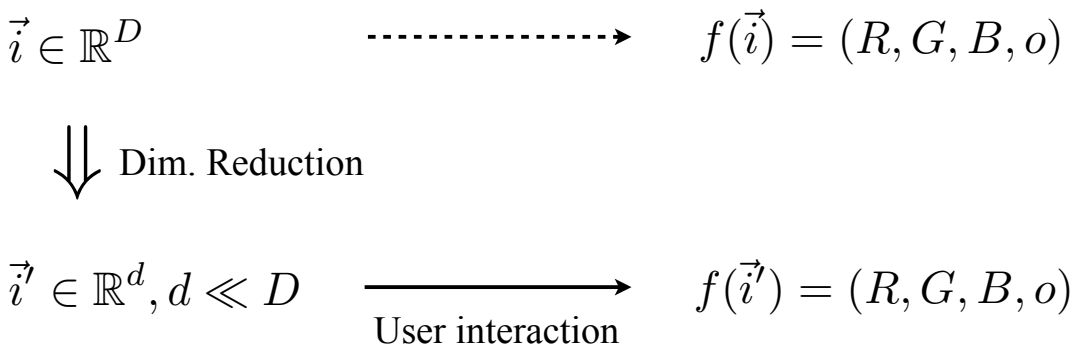


Figure 5.1: The diagram illustrating how dimensionality reduction algorithms help to create multi-dimensional transfer function for multi-channel data.

users need to see the entire distribution to find out areas of interest. By exploring the transfer function domain with special widgets, users can get insights about the data. However, as shown in the first row in Figure 5.1, the dimensionality of the original domain D is over what human can see. Therefore, defining a transfer function with the domain is limited. The dimensionality reduction algorithms reduce the original domain to a low dimensional subspace, d, and users can explore the reduced domain for it is in 2 or 3 dimensional space. With the reduced domain, users can define a transfer function through trial-and-error user interaction as usual. The expressiveness of function $f(\vec{i'})$ is preserved as much as the amount of variance preserved by dimensionality reduction algorithms.

5.4.1 Reduced Transfer Function Domain

Our goal in this work is to support the design of multi-dimensional transfer functions for multi-channel data sets. We define multi-channel data to be a set of sample data from the same object but with different types of measurement methods as described in Section 5.5.1. With multi-channel data, the areas scientists are most interested in, are often described in complex ways: for example, areas where first and third channels are active together and where the activation area is around the boundary of an object. This type of area cannot be found with traditional 1D transfer functions because assigning color with regards to only the intensity of a single channel cannot capture the correlations between multiple channels.

The first step of our transfer function design process is collecting features of voxels. We construct feature vectors defined for each voxel to contain enough information to encode areas of interest. Two sets of data are prepared for our experiments. The first data set, which we refer to as *24D Tissue Edge Data*, is as the name indicates, confocal volume image data of the edge of a section of brain tissue. In this case, the contrast between the tissue with three spectrally distinct fluorescent labels specific for three proteins or nucleic acids and often displayed with three primary colors and the background containing no fluorescent labels is very high. In this data set, the feature vectors consist of 24 fields in our example. The first three fields are intensity information for each channel, (u_1, u_2, u_3). The next information included is gradient information to isolate voxels near boundaries. Because we are not restricted to the size of feature vectors, which is one of the strengths of our approach, we add not only gradient magnitude, but also each component of the gradient vector, i.e. $\left(\frac{\partial u_i}{\partial x}, \frac{\partial u_i}{\partial y}, \frac{\partial u_i}{\partial z}\right)$ for $i = 1,2,3$. In addition, discrete Laplacian values are appended to capture ridges and valleys as in Ref. [], i.e., $\frac{1}{6}\left(\frac{\partial^2 u_i}{\partial x^2} + \frac{\partial^2 u_i}{\partial y^2} + \frac{\partial^2 u_i}{\partial z^2}\right)$ for $i = 1,2,3$. Finally, two values from first order statistics are added: mean and variance of channel intensity. In our example, 3x3x3 voxel cubes are used to capture textural characteristics [CR08].

One may argue that the feature vector is superfluous in the 24D Tissue Edge Data, but even though it may have redundant information, the dimensionality reduction algorithms can filter out redundancies. Note also that one has all the freedom of adding or removing features; if a set of features is believed to be effective in isolation, adding to the current feature vector does not cause much overhead. If an added feature is new information, the dimensionality reduction algorithm will capture the large variance of the feature. Even if it turns out that they are not effective, the dimensionality reduction algorithm will also eliminate the small variance of the domain.

On the other hand, the second data set, which we call *6D Blood Vessel Data*, is simple and straightforward. It has only 6 fields: intensity and gradient magnitude from 3 channels. This volume data set is centered around a blood vessel in the hippocampus. This data set is to verify that our method can provide the same way as in Kniss et al. [KKH02] and with relatively smaller dimensionality. That is, once we acquire experience about which field effectively isolates areas of interest, we can carefully choose

only the fields needed for the visualization goals. By reducing the number of fields, one can reduce the computational cost and generate better result as we will describe shortly in Section 5.6.3.

The second step is to run the dimensionality reduction algorithm. The input is the set of feature vectors we constructed in the first step. The reason why we apply the dimensionality reduction algorithm is that the dimensionality of the original feature vector is too high - 24 in our example but it may be even larger depending on the visualization goals. What we get after dimensionality reduction is a compact form of a voxel distribution in 3D. Instead of working with 24D data, the dimensionality reduction algorithm enables us to stay in 3D space. The relationship between the original 24D distribution and the newly computed 3D distribution is that the 3D distribution best approximates the original data. What is preserved when transforming feature vectors from 24D to 3D, is the relative distance between feature vectors. Two vectors that are close to each other in high-dimensional space are close in low-dimensional space as well. This allows us to assign similar colors to two voxels that are located close to each other in the resulting 3D domain because they are also close in the original domain.

The final step for our transfer function design is to navigate the domain produced in the second step. This is the same procedure visualization scientists go through to create multi-dimensional transfer functions in previous work. Dual domain navigation and coloring widgets [KKH02] help examining the domain in this step. The feature domain, which is actually produced by the dimensionality reduction algorithms, is created in such a way that large variances of fields in the original data are shown in the feature domain, too. Preserving the variance of the original data helps visualization scientists to more easily identify areas of interest in the volume rendering domain.

5.4.2 Scalability

Due to recent improvements of confocal scanning technology, the size of the image data that scientists usually handle has reached giga-pixels and more. The computational cost of most steps in the dimensionality reduction algorithms, such as computing pairwise distance and eigenvectors, or running the shortest distance algorithm on billions of elements are, however, sometimes prohibitively expensive. Although there have

been many algorithmic attempts to avoid this barrier [ST03], we were unable to process very high resolution images in just a single run.

We cut out a small cube (128 x 128 x 24 voxels for 24D data and 50 x 50 x 24 voxels for 6D data) from the entire data volume, containing the most interesting parts for scientists. With the small subset of data, a transfer function can be calculated much more easily. With the transfer function, we extend the result to other parts of data with the following algorithm:

- We compute the feature vector for X, a point in the large volume, and we find 3 nearest neighbors of X, in terms of Euclidean distance in the feature vector space, from the reference data, $\{X_i\}_{i=1}^{N}$, that we used for transfer function design. Then we can represent X as a weighted sum of the three vectors $X_{(1)}, X_{(2)}, X_{(3)}$ with the optimization problem:

$$\text{Minimize } |X - \sum_{i=1}^{3} w_i X_{(i)}|^2$$
$$\text{with respect to } \sum_{i=1}^{3} w_i = 1$$

- We estimate the low-dimensional embedding of X by interpolating the three neighbors. We use the same assumption that the Locally Linear Embedding algorithm [RS00] makes; if X and its neighbors are close enough, then we can assume that X is on the linear plane that its neighborhood points generate. We use the weights w_1, w_2, and w_3 to get the low dimensional coordinate, Y, of X by linearly interpolating from $Y_{(1)}, Y_{(2)}$, and $Y_{(3)}$.

Another scalability issue is for LLE. LLE requires computing the eigenvectors that have the smallest positive eigenvalues. Although the rest of the algorithm is relatively very simple and does not require solving dynamic programming as in Isomap, we failed to compute the eigenvectors of 400,000 x 400,000 matrix. Therefore, if one would to use high resolution images as the reference image for transfer function design, LLE may not be the best choice for the work. However, computation of 60,000 x 60,000 matrix took a significant less time thanks to a efficient numerical method [Fri02].

5.5 Experiment

We first introduce the volume data we used for our experiment in this section. Then we present 3D transfer function and direct volume rendering results, highlighting areas of interest, generated in the transfer function domain.

5.5.1 Light Microscopy Volume

Here we apply our new transfer function design process to a data set of images acquired by confocal microscopy of immunolabeled mouse brain tissue. The tissue is labeled with 3 different fluorescent antibodies to localize three proteins within the tissue. The image consists of three channels, each of which is the emission of a different wavelength of light collected from these fluorophores within the specimen. This data set was generated at National Center for Microscopy and Imaging Research (NCMIR) and deposited into the Cell Centered Database (CCDB; http://ccdb.ucsd.edu, accession # MPID6674), a resource for imaging data sets from advanced light and electron microscopies [MGW$^+$02a].

The original microscopy product is a mosaic of 176 overlapping image stacks that were acquired and assembled into one mosaic image encompassing the entire hippocampus region of the mouse brain, acquired using a laser scanning confocal microscope with a precision motorized stage. The original wide-field mosaic image can be browsed at multiple resolutions: the entire specimen, regions of interest, certain cell populations within regions, and even cellular compartments (at maximum resolution). This large-scale imaging technique is used to scan large regions of the mouse brain to look for expression of the pannexin1 protein, a recently discovered protein that is thought to be an ATP release channel participating in ATP signaling cascades. It is highly expressed in the brain. Because the brain is a highly heterogeneous organ and expression levels and patterns can vary within sub-domains, even within the same class of cell, these brain mosaics span an expanse of millimeters of tissue with each tile is recorded at close to the highest resolution achievable by light microscope. In these images, Pannexin1 protein expression (green) is examined in comparison with that of gap junction protein, connexin43, (red) and the astrocyte marker, glial fibrillary acidic pro-

tein (GFAP) (blue). The aim is to learn where pannexin1 protein is being utilized in brain tissue at multiple levels of resolution and determine what other cellular partners might bind to the Pannexin1 protein assemblies.

This image data is rich in content and large in size. Volumes like these usually contain much more information beyond the original experimental intent. They are good source material for data mining and sharing technology that stimulates new avenues of investigation. These large-scale mosaic image volumes must be cropped to a manageable size at an appropriate scale in order to demonstrate the findings for collaborators and in publications such as the examples in Figure 5.3 and Figure 5.4. A big challenge is presenting this data set in a way that retains both its complexity and protein specificity to optimally convey any novelty and significance within its biological contexts.

5.5.2 Experiment Design

We seek three results from the experiment: 1) rendered image quality, 2) the distribution and shape of transfer function domain, and 3) eigenvalue spectrums of PCA and Isomap algorithms. We evaluate the image quality of various images generated by transfer function. Since there is no quantitative way to compare the quality of two images generated from two different transfer function domains, we show several rendered images that successfully isolate features of interest. 1D transfer function domain or even 2D domain is often not able to isolate features. Only the domain having features spread out nicely in it can provide a way to highlight those features. For the distribution and shape of transfer function domain aligns with the first result. We will show the domain generated by dimensionality reduction algorithms and see if the shape is nicely stretched in the 3D space. If points are located close to each other, it is more likely for points from different areas of interest to be overlapped together. If this happens, it is impossible to isolate two areas. As we will discuss in Section 5.6.3, domain from LLE and other algorithms can only be compared by this result. The last result we will present, eigenvalue spectrums, compares the quality of PCA and Isomap. These two algorithms produce a list of eigenvectors and corresponding eigenvalues. We are interested in three largest eigenvalues and if those three dominate the spectrum, it indicates that we have achieved small variance loss during the reduction.

5.6 Result

We present results from the experiment we designed in Section 5.5. First, we show the transfer function domains obtained from the different dimensionality reduction algorithms, and we present rendered images of 3D microscopic data, showing highlighted features. Finally, we discuss the difference between the three algorithms.

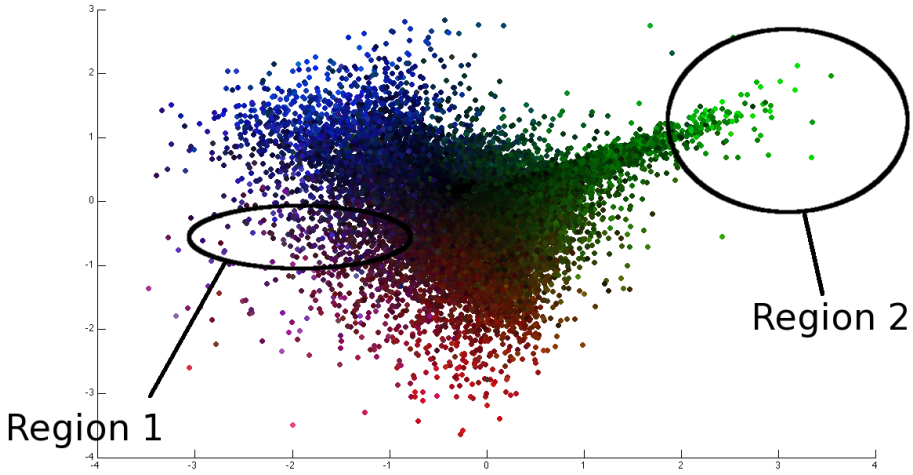
5.6.1 Transfer Function Domain

Figure 5.2 shows the domain of a transfer function of the 24D Tissue Edge Data described in Section 5.5.1. Figure 5.2(a) is the view from the top of the domain, i.e., from the positive infinite direction of the Z axis, or projection to the XY plane. Figure 5.2(b) is the view from one side of the domain, that is, from the positive infinite direction of the Y axis, or projection to the XZ plane. This is produced after applying the Isomap algorithm with 500 landmarks with $K = 50$ nearest neighborhood searchings. To give basic information about the distribution, the original channel intensities of the voxels were initially assigned to points.

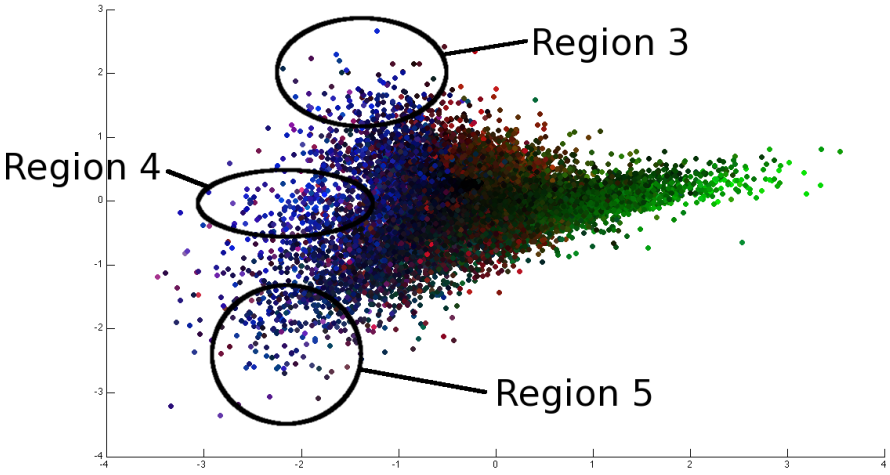
In Figure 5.2(a), three channel intensity values are clustered at each corner. Voxels expressing two channels together are between two corners. Especially region 1 denotes where astrocytes and Connexin43 proteins are expressed around the boundary of the cell. The distribution of voxels that express Pannexin1 protein branches out from the center, region 2. The edge of the branch is the voxels representing blobs of Pannexin1 as shown in Figure 5.4(a).

It is more clear to see how Connexin43 proteins are distributed in the sample with Figure 5.2(b). The voxels are spread according to ridges and valleys; region 3 is where valleys going down to +z depth from right to left; region 4 is where ridge voxels are distributed; and region 5 denotes another valleys but the opposite direction to region 2.

Many widgets for editing transfer functions, e.g., as in Kniss et al. [KKH01], can be used in the same way with this domain. For the images in Section 5.6.2, only a Gaussian widget was used. The Gaussian widget in the 3D domain maps point x to a



(a) View from Top



(b) View from Side

Figure 5.2: The domain distribution of feature vectors after Isomap dimensionality reduction. Figure 5.2(a) is the view from the top of the domain, i.e., projected to the XY plane. Figure 5.2(b) is the view from a side of the domain, i.e., projected to the XZ plane.

color according to the Gaussian distribution:

$$I(x;\mu,\sigma) = \frac{1}{\sqrt{2\pi}|\sigma|^{3/2}} \exp\{-\frac{1}{2\sigma^3}(x-\mu)^T(x-\mu)\} \tag{5.2}$$

where μ denotes the location of the center, and points around the center are affected by the widget. σ decides how far from the center the widget's influence reaches. Larger σ values color a wide range of points whereas smaller σ affects only a small range of points. The location of the center, μ, is selected by users, and our tool provides a method to adjust σ. In the future, we may investigate a tool which separately controls σ in three directions, but this widget was effective enough in our experiments.

5.6.2 Rendering Result

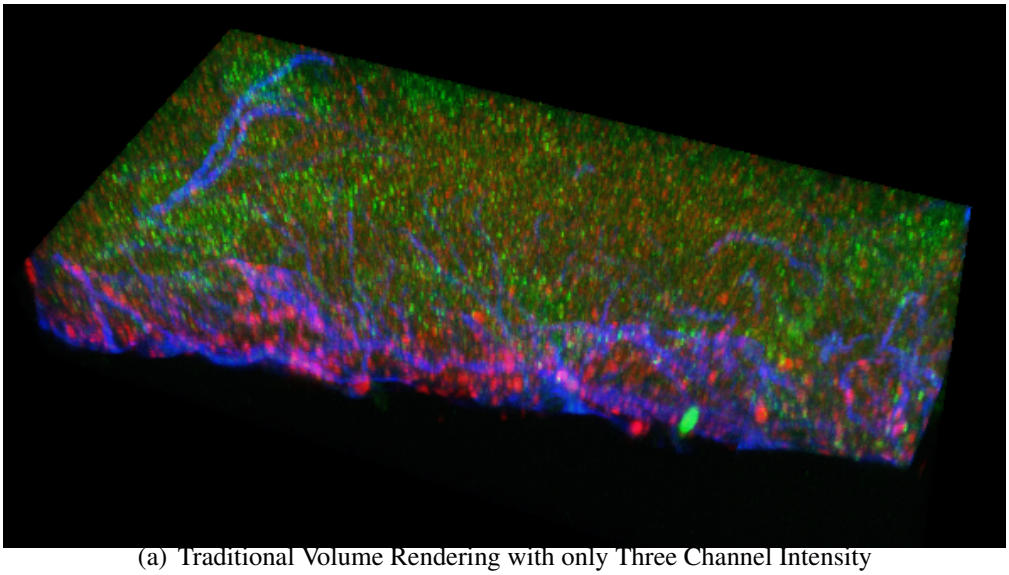
We rendered two data sets, 24D Tissue Edge (512 x 256 x 24 voxels) and 6D Blood Vessel (300 x 250 x 24 voxels), with DeskVOX [Sch11], an open-source volume rendering software. A maximum intensity projection algorithm was used for blending the slices. Figure 5.3, Figure 5.4, and Figure 5.5 shows several screen shots.

Tissue Edge

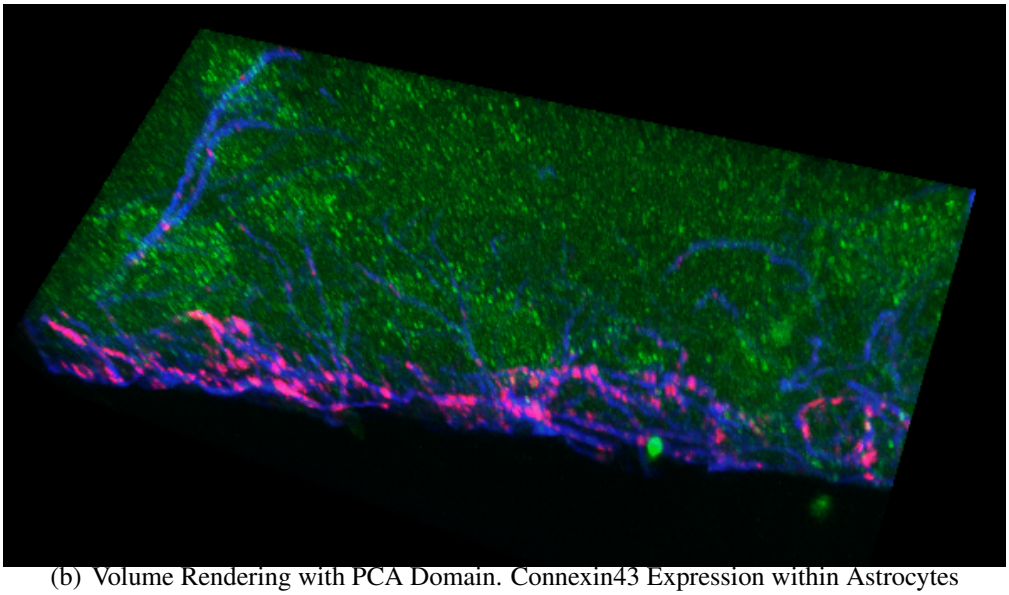
In Figure 5.3(a), a small region of the three-color image shows the edge of the hippocampus tissue with triple fluorescent immunolabeling (Green=Pannexin1 protein, Red=Connexin43 protein, Blue=GFAP within astrocytes). In Figure 5.3(a), a projection of the original multi-channel RGB image stack is displayed for comparison with improved visualizations of key elements generated using our new transfer function design process that are shown in Figure 5.3(b)-5.4(b).

The astrocytes (blue in Figure 1(a)) cover and their processes wrap the outer surface of the hippocampus. These astrocytes, as well as other cell types, express Connexin43 (red). Figure 1(b) nicely highlights the Connexin43 (red) expression within astrocytes (blue) surrounding the hippocampus tissue while suppressing the Connexin43 labeling that is present among unlabeled, and therefore unidentifiable, cells deeper within the tissue slice.

There are aspects of Pannexin1 expression within this image region that are only apparent after enhancement in Figure 5.4(a) (orange) including medium sized round ar-

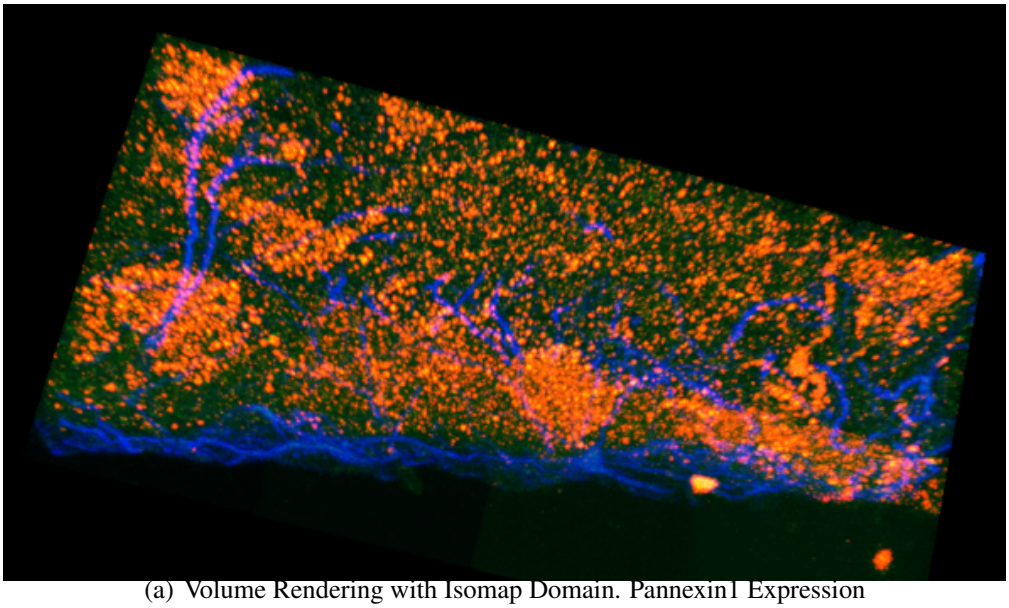


(a) Traditional Volume Rendering with only Three Channel Intensity

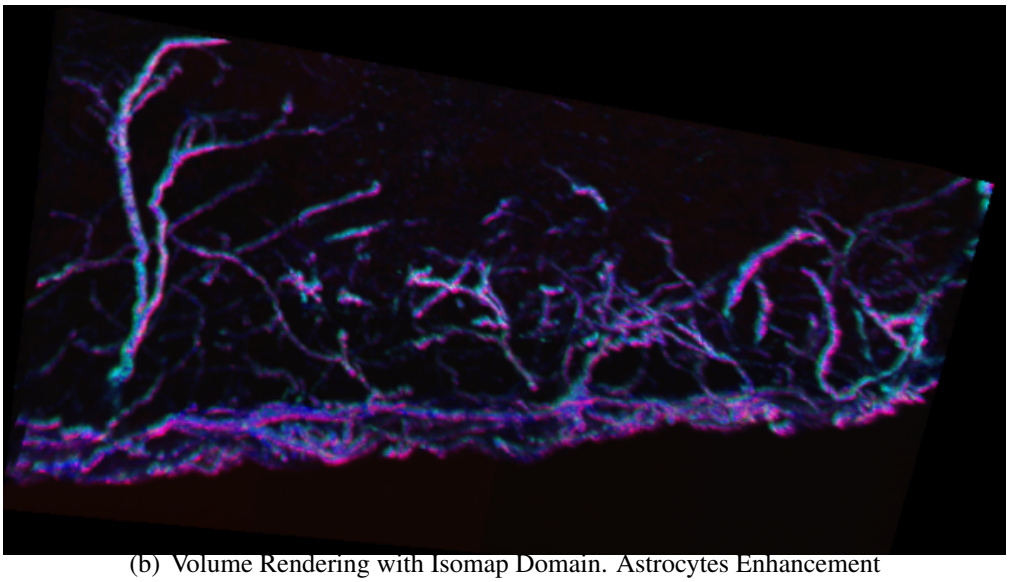


(b) Volume Rendering with PCA Domain. Connexin43 Expression within Astrocytes

Figure 5.3: Images of a three-channel 512 x 256 x 24 volume. All the images depict the same volume but with different transfer functions. Maximum intensity projection was used for slice compositing. Figure 5.3(a) is a result with a transfer function that maps the three data channels to red, green, and blue, respectively. Figure 5.3(b) shows examples of our methods. The image was generated in the domain in which our dimensionality reduction algorithms were applied. Figure 5.3(b) shows the area where Connexin43 and astroctytes are both expressed at the boundary.

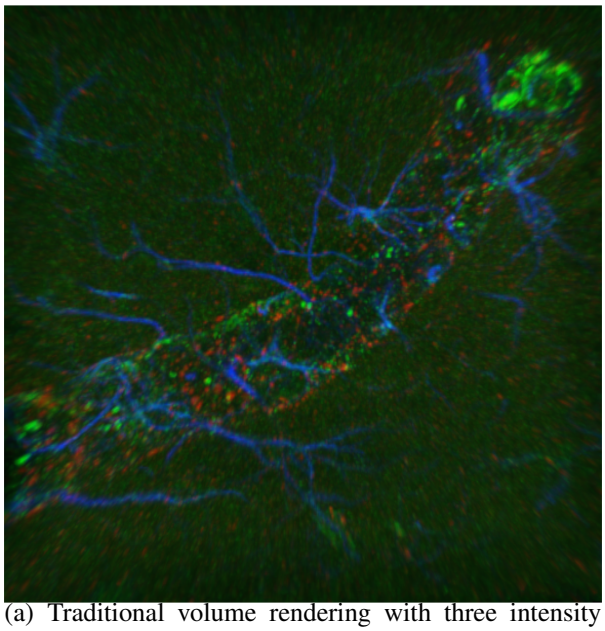


(a) Volume Rendering with Isomap Domain. Pannexin1 Expression

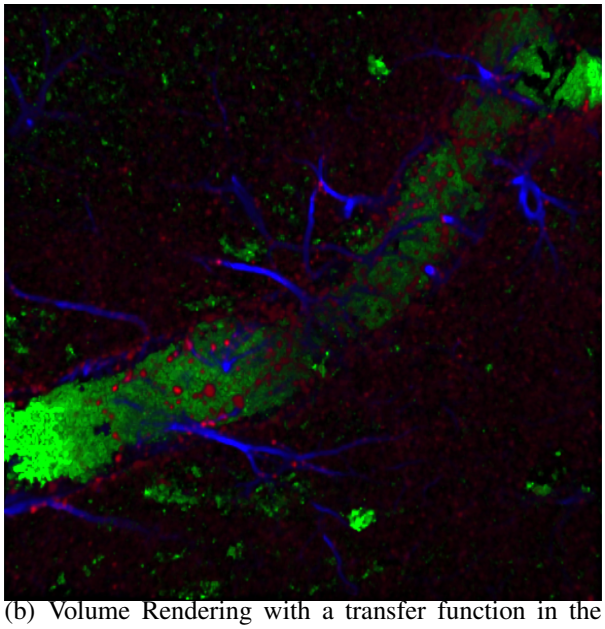


(b) Volume Rendering with Isomap Domain. Astrocytes Enhancement

Figure 5.4: Images of a three-channel 512 x 256 x 24 volume. All the images depict the same volume but with different transfer functions. Maximum intensity projection was used for slice compositing. Figure 5.4(a) and 5.4(b) show examples of our methods. Both images was generated in the domain in which our dimensionality reduction algorithms were applied. Figure 5.4(a) emphasizes where only Pannexin1 is expressed. Figure 5.4(b) shows where astrocytes with valleys are highlighted with different colors.



(a) Traditional volume rendering with three intensity channels only



(b) Volume Rendering with a transfer function in the LLE domain

Figure 5.5: Rendered images of a three-channel 300 x 250 x 24 volume. The figure on the top shows the original image that maps three intensities to red, green, and blue. The bottom figure shows the rendered image with a multi-dimensional transfer function. In this figure, the blood vessel is highlighted in green whereas the original green intensity shown on the top is ignored. By turning off the original green channel, one can more clearly how the red and blue channel interact each other around the blood vessel.

eas with more dense labeling that may be cell bodies whose cell type could be identified with future immunolabeling experiments. Similarly, “squiggles" of more dense Pannexin1 expression can be identified on the right side of Figure 5.4(a) only that they are likely cellular extension coming from these cell bodies and penetrating deeper into the hippocampus.

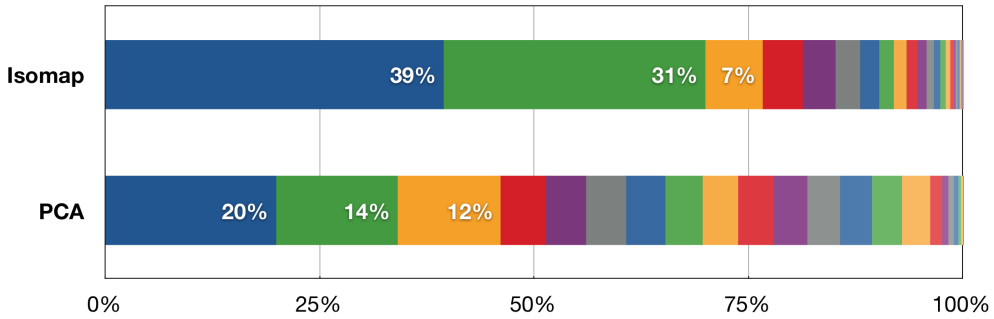
Figure 5.4(b), is an enhancement of the astrocyte cell labeling alone (blue) that highlights their structure. Not only do they cover the outer surface of the tissue, but in Figure 5.4(b) the contour of these glial cells shows how they also send branching projections deep into the hippocampus in order to establish contact with numerous other cell types known to include neurons. These branching interior projections are extremely difficult to make out in the original image shown in Figure 5.3(a) and cannot be demonstrated in this way using traditional techniques of adjustments of RGB levels.

Blood Vessel

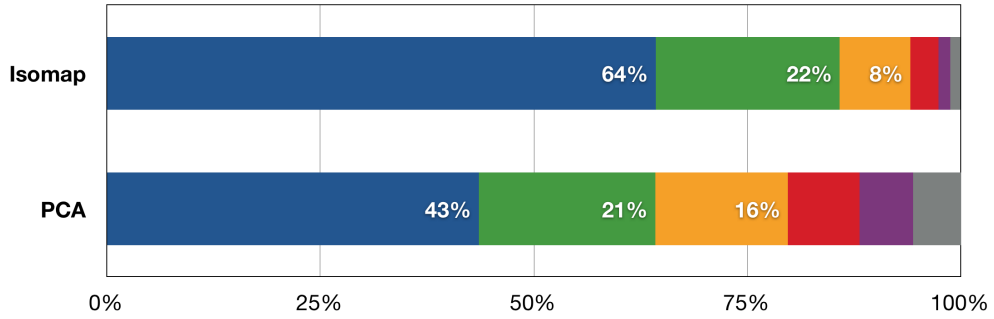
Figure 5.5 shows the blood vessel in a hippocampus. Figure 5.5(a) is from the traditional volume rendering with 1D transfer function, i.e. assigning red, green, and blue to each channel, respectively. It is hard to recognize how astrocyte and Connexin43 gap junctions interact each other around the blood vessel mainly due to tissue complexity. Furthermore, because there is no stain inside the blood vessel, it is difficult to recognize where the vessel actually is located.

On the other hand, Figure 5.5(b) is an enhancement of Figure 5.5(b) using our method. During the transfer function design, we colored the area where three intensities and three gradient magnitude values are all near zero. That area in the transfer function domain successfully isolated the inside of blood vessel.

Improving the process of modeling this 3-color protein-labeling data so that it can be meaningfully displayed, annotated and queried by other researchers with the techniques discussed here will help the CCDB integrate the critical function of data management with data mining tools for investigating the structural variation and molecular complexity within the nervous system.



(a) Eigenvalue Spectrum for 24D Tissue Edge Data



(b) Eigenvalue Spectrum for 6D Blood Vessel Data

Figure 5.6: Eigenvalue spectrum from Isomap and PCA. Each color corresponds to one of the eigenvalues, and its size represents its relative magnitude. Larger percentage of first three largest eigenvalues (three from the left) indicates that the three corresponding eigenvectors can better explain (approximate) the original data. With the 24D Tissue Edge data, the three eigenvalues of Isomap sums up to 75.4% and PCA reaches up to 46.0%. With the 6D Blood Vessel Data, it showed a much better spectrum, 94% with Isomap and 80% with PCA.

5.6.3 PCA vs. Isomap vs. LLE

A common way of measuring the quality of a low-dimensional representation is to look at the spectrum of eigenvalues [LHWS08] in PCA and Isomap. Figure 5.6(a) and Figure 5.6(b) show the eigenvalue spectrum of two data sets, 24D Tissue Edge and 6D Blood Vessel data, respectively. Each bar represents the eigenvalue magnitude, and they are sorted from largest to smallest. The first spectrum is the result from 6D space and 6 eigenvalues are generated. Similarly, the 24D Tissue Edge data has 24 eigenvalues. As we project high-dimensional data into 3D space, the entire data is approximated with only three eigenvectors. A large eigenvalue means that the corresponding eigenvector can better explain the original data, and 100% means it can reconstruct the original data without any error.

Figure 5.6(b) shows that the sum of three largest eigenvalues produced by Isomap is 94%, which dominates the entire spectrum. Only 6% of the total variance is lost by the dimensionality reduction. In this case, even with the first and second largest eigenvalues, it sums up to 86% and then with this result, one can reduce the dimensionality down to 2D, which is much easier to work with. The spectrum of PCA was not as high as that of Isomap, namely only 80%. However, compared to the 24D Tissue Edge data, it is a promising result. As shown in Figure 5.6(b), the sum of the three eigenvalues from PCA achieved only 46.0% of that of the total eigenvalues. The spectrum of Isomap shows that the top three eigenvalues dominate the entire spectrum, 75.4%.

We believe that the difference between the 6D Blood Vessel and 24D Tissue Edge data stems from the characteristics of the data. The 24D Tissue Edge data contains more mathematical feature information than 6D Blood Vessel data. Some of them may contribute to identifying features that we are interested in, but the rest may not have played a significant role. Adding more features, or increasing the dimensionality of the domain, makes all three algorithms difficult to reduce to a lower dimension. It is mainly because a new feature spread out the domain without the feature in the new axis. Then the new domain with the new feature has more variance in the distribution than the one without it.

However, we were unable to quantitatively compare LLE with PCA and Isomap. It is because LLE generates the result from a different method, finding the smallest

eigenvalues. While PCA and Isomap approximate the high-dimensional vector by cutting off eigenvectors from the observation that eigenvectors with smaller eigenvalues contribute less to the reconstruction of the original vector. However, the eigenvalue spectrum in LLE is not used for this type of approximation, and thus comparing the magnitude of eigenvalues does not reveal any information to us.

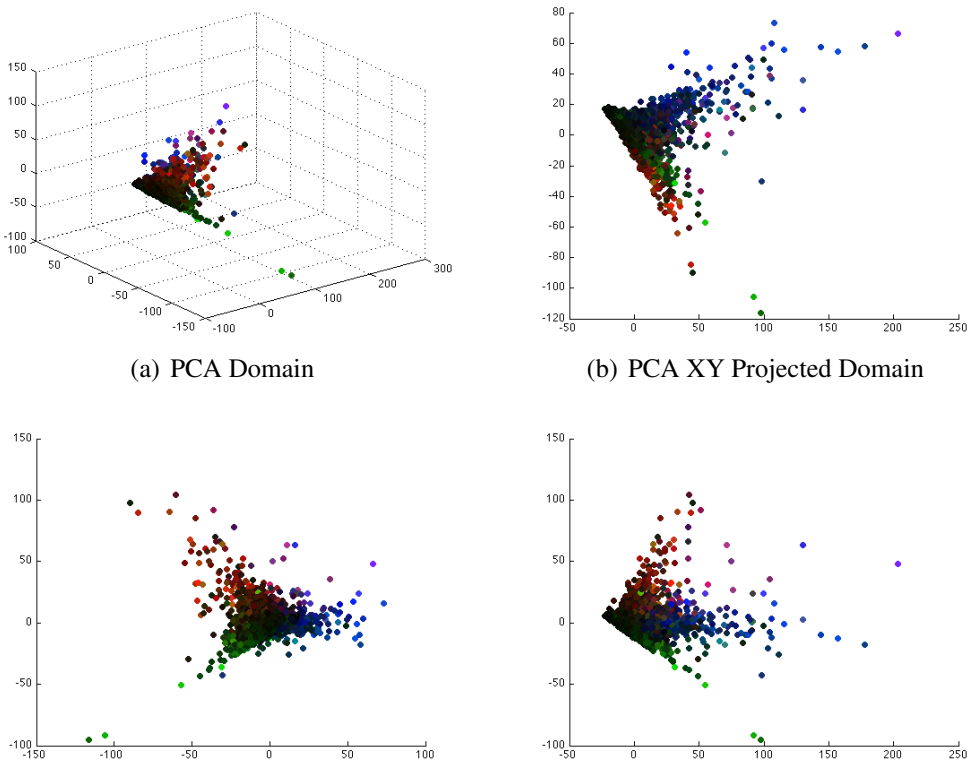
There have been many reports on the comparison between algorithms with predefined data [ST03, DG03b] but none of them was able to compare quantitatively. It is not clear how to measure "how faithfully" the low dimensional representation explains the original data. Hence, the only way that we were able to use for the comparison of the result of PCA, Isomap, and LLE was a qualitative way, i.e. by looking at the domain itself.

Figure 5.7 and Figure 5.8 list the two domains generated from 6D Blood Vessel data with PCA and LLE. To help the reader's understanding of the point distribution, we colored them with gradient magnitude of three channels. Furthermore, because it may not clear to understand the shape of 3D distribution, we also list 2D projection of the domains to three major planes, XY-, YZ-, and ZX-plane.

Both PCA and LLE were able to cluster voxels at three corners, each of which indicates high gradient values of one channel. The shape roughly looks like a tripod. The shape of PCA domain is simple, many near-zero gradient magnitude values are densely clustered and it lays out points too closely to each other, resulting distinct areas of interest to display identical colors when a Gaussian widget selects a region in the PCA domain. On the other hand, LLE domain spreads out the distribution more clearly, which makes the dual navigation easy.

5.7 Further Applications

The result of our dimensionality reduction algorithms can be used as a starting point for trial-and-error [PLS+00] approaches. One can explore the result with the same set of widgets developed for dual domain interaction. This means that we do not generate one final transfer function. We only give a compact form of the domain. Visualization scientists can start with the compact representation that can still effectively isolates



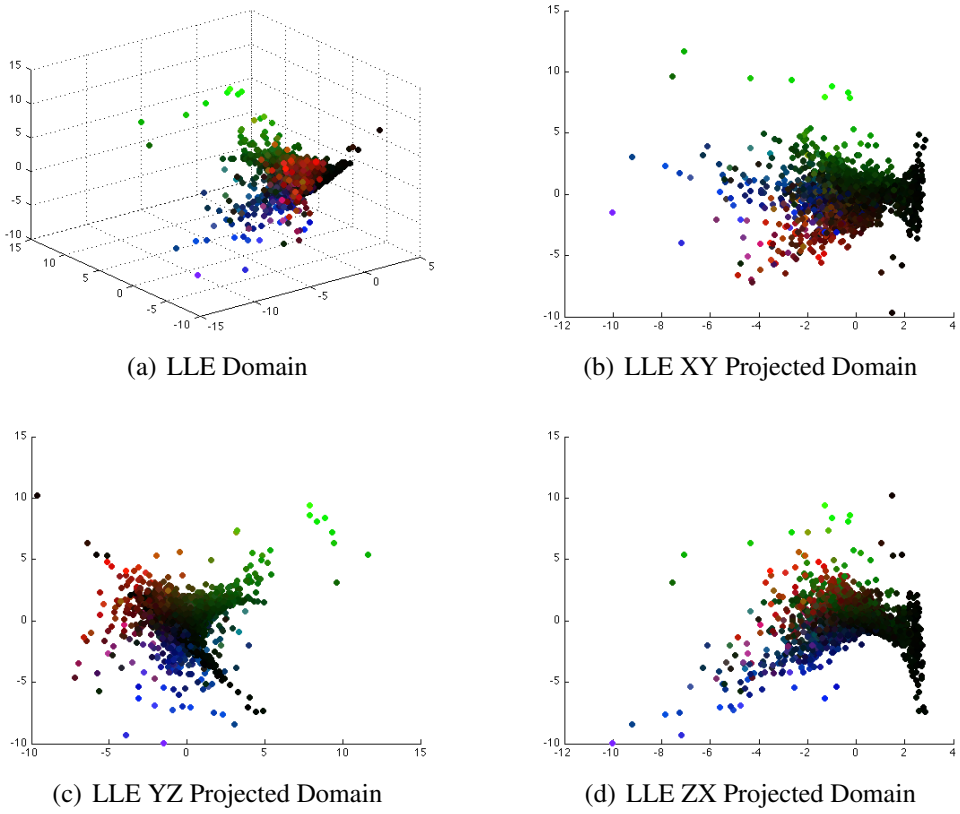
(a) PCA Domain

(b) PCA XY Projected Domain

(c) PCA YZ Projected Domain

(d) PCA ZX Projected Domain

Figure 5.7: PCA domain after reducing the dimensionality of feature vectors into 3D: Figure 5.7(a) shows a 3D view of the domain and Figure 5.7(b), 5.7(c), and 5.7(d) show the projected domain onto the three main axes.



(a) LLE Domain

(b) LLE XY Projected Domain

(c) LLE YZ Projected Domain

(d) LLE ZX Projected Domain

Figure 5.8: LLE domain after reducing the dimensionality of feature vectors into 3D: Figure 5.8(a) shows a 3D view of the domain and Figure 5.8(b), 5.8(c), and 5.8(d) show the projected domain onto the three main axes.

areas of interest. Many algorithms in transfer function design are geared towards better recognition of features in the data. Gradient information is useful because it can capture the boundary of objects. The domain we get after applying Isomap contains enough data to successfully isolate features - otherwise, we can simply add more features that effectively isolate interesting parts in the image before applying dimensionality reduction algorithms. Then, the dimensionality reduction algorithms can strip out superfluous components in the feature vectors. Thus, the domain represents the best possible low representation in 3D, especially compared to those with only three types of information.

In addition, if advanced interfaces for classification or automatic transfer function generation algorithms [HHKP96, TLM03, PF08] are changed to accommodate the domain created here, they might produce better results than using channel intensities only. Furthermore, the low-dimensional representation (3D) is much smaller than the original dimensionality (24D in our example), and thus the space and time complexity for handling this smaller data set can also be kept lower. One advantage over other algorithms is that dimensionality reduction algorithms are unsupervised learning algorithms [DHS00], that is, they do not require user input during the computational process, nor prior knowledge about the data from the users, such as pre-classification of the input data or selection in stochastic processes.

Finally, the low-dimensional representation can be further processed for clustering [MWCE09] to guide visualization scientists. Clustering the domain gives scientists a rough guideline for where to start. This idea strengthens our approach because the direction of the axes in the reduced domain has no specific meaning. Each axis of the original high-dimensional data represents one data channel. The axes help to understand the correlation or trends between multiple features. Without this information, it may not be easy to understand which part in the 3D domain represents which features. The clustered result will alleviate this inherent limitation of dimensionality reduction as it gives an initial starting point.

5.8 Summary

We proposed a new method to build transfer functions for multi-channel data sets. Feature vectors consist of multiple mathematical properties, as well as each of the channel intensities. The high dimensionality of the feature vector does not allow a user to directly visualize the transfer function domain. We applied two recent developments in nonlinear dimensionality reduction, Isomap and LLE, and a classical linear algorithm, PCA, to reduce the high-dimensionality to a manageable size. The algorithms guarantee that the low-dimensional domain retains details of the high-dimensional data set. The major benefit of looking at combined features is that co-occurrence, either a positive or negative correlation, of multiple components in the feature vector can have a different color by isolating the regions from other parts.

We believe our approach is general enough to be applicable to other kinds of data domains beyond the neuroscience imaging examples we show here [PLS+00]. Medical imaging communities merges multiple data sources such as MRI or CT and the multi-dimensional transfer function originally came out of MRI data sets. If one wants to consider multi-modal data and to design a transfer function with multi-dimensional domain, for example intensity and gradient maginitude, the dimension of the total features considered easily exceeds 3D. Our approach can be applied in the same way as we discuss in this chapter but it would be interesting to study our approach in depth on these data sets.

While Isomap and LLE already showed sufficient results for our goal, we are aware that many algorithms have been proposed for nonlinear dimensionality reduction since the two seminal works, Isomap and LLE, published in 2000. It would be worth investigating other variants [BN03, DG03a] to test whether one of these variant algorithms can produce better results.

Finally, although the dimensionality algorithms run only one time, the amount of computation needed for these algorithms remains a big challenge. However, we expect that further optimization of the algorithms and efficient implementations will alleviate this problem. For example, general purpose GPUs can be used to accelerate parallelizable parts and the use of sub-sampled data to find low-dimensional representations, along with a reasonable reconstruction algorithm, should improve overall computational

cost.

Acknowledgement

This chapter, in part, has been published as “Dimensionality reduction on multi-dimensional transfer functions for multi-channel volume data sets” by Han Suk Kim, Jürgen P. Schulze, Angela C. Cone, Gina E. Sosinsky, Maryann E. Martone in Information Visualization Journal, Palgrave Macmillan in 2010 [KSC+10a]. The dissertation author was the primary investigator and author of this paper.

This chapter, in part, has been published as “Multichannel transfer function with dimensionality reduction” by Han Suk Kim, Jürgen P. Schulze, Angela C. Cone, Gina E. Sosinsky, Maryann E. Martone, in the proceedings of SPIE Visualization and Data Analysis conference in 2010 [KSC+10b]. The dissertation author was the primary investigator and author of this paper.

Chapter 6

Viewpoint Selection

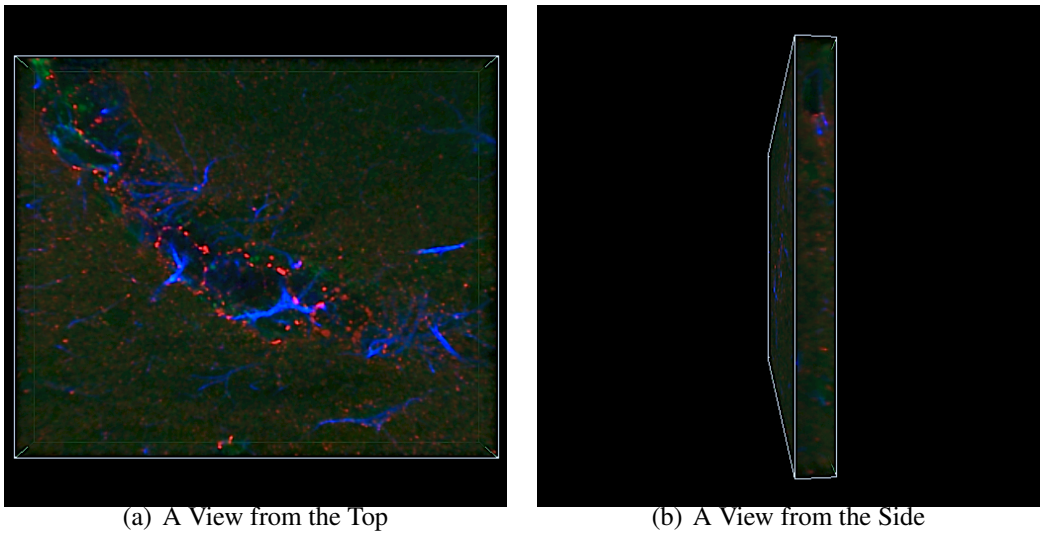
In the previous chapters, we have discussed the role of transfer function and presented a new method to overcome the limitation of transfer function exploration with multi-channel data. In exploring multi-channel data, multi-dimensional transfer function greatly helps users to understand the data. However, because of the ability to dramatically change the look of a volume, it is often necessary for users to rotate the volume to understand the effect of a newly set transfer function. By rotating the volume, users try to find a good viewpoint that explains the volume well, and by doing so, they can understand the current setting of transfer function. In addition, most volume rendering systems from multi-channel data provide a way to turn on and off each channel separately to focus on one or two channels. With this type of exploration, it is also desirable to find a good viewpoint to display the volume effectively.

Therefore, this chapter discusses another aspect of visual exploration in volume rendering: viewpoint selection algorithm. We first define the viewpoint selection problem and explain the importance in volume exploration. Then, we review previous approaches to this problem, and present their limitations.

6.1 Problem Formulation

Volume rendering is a process of mapping a 3D object to a 2D image. The process inherently maps multiple points in the 3D object to the same image pixel, which we call occlusion. The amount of occlusion depends on where the camera is located.

The following figure shows one example.



(a) A View from the Top (b) A View from the Side

Figure 6.1: Two volume rendering images from different viewpoints. The volume is a thin slab of a biological sample. If the viewpoint is set as "from the top," the final image reveals all the details of the volume. However, if the volume is projected to the side of the volume, the information that users can get is very limited.

Figure 6.1 shows two rendered images of the same biological sample from two different viewpoints. The volume is a thin slab showing blood vessels crossing diagonally and proteins wrapping around the vessel (colored in red and blue) in Figure 6.1(a). In Figure 6.1(a), the viewpoint, i.e. the location of the camera, is at the top of the volume. From this view, we can see all the details in the data, such as proteins expressed in red and blue, and it is easy to understand the geometry of the volume itself. The image from this viewpoint does have occlusion, but, because the z-depth is small, the amount of occlusion is limited. On the other hand, Figure 6.1(b) shows the volume from the side. In this view, it is hard to see the details in the volume because the amount of occlusion is much bigger than Figure 6.1(a). These two figures show the importance of the amount of occlusion and the viewpoint for volume rendering in understanding the data effectively. Therefore, positioning the camera in a "good" place is critical in improving user experience.

However, if the camera can be located at any location in 3D space, there are three parameters to specify one location: azimuth, elevation angle, and distance from

the center. In order to keep the number of parameters reasonalbe, we make the following assumptions:

Viewing Sphere We assume that the viewpoint is located in a viewing sphere [BD90]. The camera can be positioned anywhere on the viewing sphere. Moving toward or away from the viewing object scales the contents in the image up or down respectively. The geometry shown in the rendered image, however, does not change.

Centered Object We also assume that the center of the object is located at the center of the viewing sphere. We do not consider the case where the object is translated away from the center.

Therefore, viewpoint selection is a problem of finding a "good" viewpoint on the viewing sphere that describes the volume effectively. One issue is that the "goodness" can be defined in many ways. The definition of a good viewpoint has a long histrory across many different disciplines [BTB98, BD90, KD76, KD79, PRC81].

Canonical view [PRC81, BTB98] is an early work on defining good views. Palmer et al. [PRC81] did experiments to rate the quality of viewpoints, finding out that participants in the experiments preferred off-axis views, i.e., a three-quarter view which makes a large number of surfaces visible. Blanz et al. [BTB98] further studied this concept with psychophysical experiments. The findings in their work also comply with the results of Palmer et al. [PRC81]: 1) most participants preferred off-axis views to straight front and side views; 2) occlusions were avoided; 3) preferred viewing directions tend to be relatively low above the ground; and 4) preferred views for familiar objects are closely aligned with the gravitational upright.

Another attempt to define a good viewpoint is aspect graph [KD76, KD79]. It organizes *general view* as a node of the graph and *visual event* as an edge of the graph. The general views are a region of views where small changes in the view direction incur changes in the geometry of the rendered image. The concept of viewing sphere here is used in almost all view selection algorithms, and there have been many studies on this topic [BD90].

Blanz et al. [BTB98] defined the best viewpoint as one from which users would take a photograph if they planned to use the photograph to illustrate the object in a

brochure. This statement means that the viewpoint helps users to understand the object only by the image from the viewpoint. Previous approaches try to achieve this goal, but the ways they approached are all different. In the next section, we will discuss how previous work translated this high-level goal into more concrete definitions of "good viewpoint."

6.2 Previous Approaches

There have been many attemps to solve the viewpoint selection problem by using information theory frameworks [CT91]. In information theory, the amount of uncertainty contained in a discrete random variable X defined on a set of events $x_1, \ldots, x_n$ is defined as

$$\mathbb{H}(X) = -\sum_{i=1}^{n} p_i \log p_i \tag{6.1}$$

where p_i is the probability for x_i. Note that $\mathbb{H}(x)$ is maximized when $p_i = 1/N$ for all i.

This information theory framework has been used in many previous approaches [VFSH01, VS03, PPB+05], and the main differences between them are on what domain of objects they define the probability distribution and how to define the distribution on each object.

Vázquez et al. [VFSH01] define a good viewpoint as one that provides a high amount of information for the scene created from the viewpoint. This approach applies the information theory described above to measure the amount of information contained in a scene. The information that the algorithm is interested in is the projected area and the number of faces seen. The domain of the probability distribution is all the faces in the 3D mesh. The probability distribution for the entropy function is defined for each face of the 3D mesh as follows:

$$\mathbb{H}(X,V) = -\sum_{i=0}^{N_f} \frac{A_i}{A_t} \log \frac{A_i}{A_t} \tag{6.2}$$

where A_i denotes the projected area of i-th face in the mesh data and A_t is the total projected area. The scene generated by the viewpoint V contains N_f many faces. The

algorithm first samples a set of candidate viewpoints from the viewing sphere. For each viewpoint, it evaluates $\mathbb{H}$. The best viewpoint is defined as the one that achieves the maximum value of $\mathbb{H}$. $\mathbb{H}$ is maximized when all the faces are visible and projected as the same size. Namely, the best viewpoint defined by Equation 6.2 prefers a scene with many faces visible and where the areas of all the faces are balanced with each other to an approximately equal size.

Vázquez and Sbert [VS03] extend the work of Vázquez et al. [VFSH01] by accelerating the exploration path in entropy calculation. Instead of a brute force search of samples in the viewing sphere, Vázquez and Sbert use an adaptive method to narrow down the search space.

Lee et al. [LVJ05] define *mesh saliency* to find visually interesting regions on a mesh. The mesh saliency is computed with the mean curvature with the center-surround mechanism. This algorithm first computes the mean curvature for each vertex v and the Gaussian-weighted average of the curvature to cut-off around the center v. The Gaussian filtering is applied twice with two different variances, fine and coarse. The mesh saliency at vertex v is defined as the absolute difference between the Gaussian-weighted averages computed at fine and coarse scales. The saliency is computed across multiple scales, i.e., Gaussian filtering with a set of different variances, and is combined with a nonlinear normalized sum. The final saliency captures all the important features in the 3D mesh, and the best viewpoint is the one in which the salient vertices are maximally visible.

Polonsky et al. [PPB$^+$05] also discuss the best viewpoint selection in the context of 3D mesh rendering. They define multiple view descriptors, saying the best viewpoint is the one that maximizes the descriptors. The descriptors defined in their work include 1) *surface area entropy*, 2) *visibility ratio*, 3) *curvature entropy*, 4) *silhouette length*, 5) *silhouette entropy*, 6) *topological complexity*, and 7) *surface entropy of semantic parts*. Surface area entropy is identical to what Vázquez et al. [VFSH01] proposed. Visibility ratio takes into account the invisible portions to find a viewpoint that maximizes the portion of visible faces and minimizes the portion of invisible faces. Curvature entropy captures the shape complexity by considering the curvature distribution over the visible portion of the surface. Silhouette length descriptor measures the total length of all

silhouette edges on the image plane. Silhouette entropy is a combination of silhouette length and curvature entropy, i.e., it measures the entropy of all turning angles between adjacent silhouette edges. Topological complexity computes the total number of critical points, e.g., maxima, saddles, and minima, along a given direction. Those critical points are the features shown from the direction. If a viewpoint direction shows the most features, that viewpoint is optimal. Finally, surface entropy of semantic parts extends surface entropy by adding semantic information of the volume, which is explored in depth by Takahashi et al. [TFTN05].

Takahashi et al. [TFTN05] extend the work by Vázquez et al. [VFSH01] to volume rendering. They use the same surface entropy function, but the domain of the function is different because there exists no mesh face in volume data. The main idea is to extract a set of isosurfaces from the volume data in a structured grid. Each isosurface has a set of faces, and the entropy function takes into account all the faces from all the isosurfaces. Because some isosurfaces are often more meaningful than the others, the algorithm assigns weights for each isosurface. In addition, the algorithm decomposes the entire volume into multiple *interval volumes* (IV). Each IV computes the surface entropy by extracting isosurfaces. Then all the IVs are weighted by the sum of opacity values in each IV so that the entropy function can incorporate the information defined in the transfer function.

Bordoloi and Shen [BS05] define a viewpoint to be good if *important voxels in the volume are highly visible*. Therefore, they measure visibility for each voxel from each viewpoint. If the overall visibility from a viewpoint matches closely with a user-defined importance function, the viewpoint is regarded as a good viewpoint. The importance function, called *noteworthiness*, can be defined in many ways, but the authors use opacity and frequency of a voxel. If opacity at a voxel is high, or if the opacity value appears less often in the entire data, the voxel is believed to be important. Namely, for voxel j, the visual probability q_j from a viewpoint V is defined as follows:

$$q_j = \frac{1}{\sigma} \frac{v_j(V)}{\alpha_j \log(1/f_j)} \tag{6.3}$$

where $v_j(V)$ denotes the visibility of voxel j from viewpoint V, α_j the opacity value, and f_j the frequency probability of α_j in the opacity histogram. σ is a re-scaling constant

to make q_j a probability distribution. With this definition, the authors find the best viewpoint that satisfies the following two criteria:

1. The rendered image from the viewpoint has the highest information content
2. The voxel visibilities are closest to being proportional to their noteworthiness.

The searching for the best view is done by entropy function $H(V)$. The entropy function, $H(V) = -\sum_{j=0}^{J-1} q_j \log_2 q_j$, is maximized when $q_0 = \ldots q_{J-1} = 1/J$. If this is achieved, the second criteria is met because the visibility at each voxel is proportional to the noteworthiness of the voxel. Picking up the maximum entropy value among all possible choices of V guarantees that the best view has the highest information content, which is the first criteria for the best view.

In addition to the new measure, Bordoloi and Shen also propose view similarity, which measures the Jensen-Shannon divergence between samples. They use view similarity to compare view samples and to determine how two views are similar to each other. The idea of comparing samples with the measure is similar to Sbert et al. [SPFG05] where the Kullback-Leibler distance is used.

Ji and Shen [JS06] followed this line of research but proposed a new definition of a good viewpoint. Their algorithm picks a viewpoint from which important features are shown on a large area and evenly distributed. The important features they proposed are 1) high opacity, 2) salient color, and/or 3) high curvature. Namely, the algorithm prefers an image that has a large projection area, and, in the image, the opacity is evenly distributed. If the rendered object is color-coded, less frequently shown color, which they call *salient color*, is more important, and the algorithm finds a viewpoint that can show the salient color in a large area. Finally, a voxel with a high curvature value normally contains more information than one with a low curvature. The algorithm gives more weight to a viewpoint that shows, as much as possible, the high curvature area.

The major difference between Ji and Shen [JS06] and Bordoloi and Shen [BS05] is the space in which they evaluate the entropy function. Both methods are based on the information theory, measuring the complexity of given information. Bordoloi and Shen measure this information on an indivisual voxel in the volume data. The noteworthiness values do not change when the information is evaluated for a different viewpoint, but

the visibility of each voxel does change. Therefore, the visibility has to be computed for each view. Ji and Shen used a different approach. Instead of evaluating the entropy for the volume data itself, they define feature volumes in which one of three features, opacity, color, or curvature, is defined. The feature volume is rendered as in normal volume rendering. Each pixel in the rendered image contains the information that is accumulated over the ray passing through the pixel. Therefore, the probability p_i of the i-th pixel is defined as

$$p_i = \frac{I_i}{\sum_{j=0}^{n-1} I_i} \tag{6.4}$$

where I_i is the information measured at each pixel. I_i can be opacity α_i, the number of pixels of the color at pixel i $A(c_i)$, or curvature c_i. Once the probability function is defined at each pixel, the entropy for the given viewpoint can be computed. The final utility function then averages out the three entropy values with pre-defined weights. Prior knowledge about data helps to effectively determine the weights for specific data.

Mühler et al. [MNTP07] proposed a system that utilizes a viewpoint selection algorithm for intervention planning. Their viewpoint selection algorithm preprocess parameters that affect the quality of viewpoints. The parameters considered are 1) object entropy as in Ji and Shen [JS06], 2) number of occluders, 3) importance of occluders, 4) size of unoccluded surface, 5) preferred region, 6) distance to current viewpoint, and 7) viewpoint stability. The system provides a way to adjust the weights of the parameters during the navigation. The precomputation provides interactivity in finding the best viewpoint. However, as we will discuss in Section 6.3, precomputing parameters is not feasible in interactive transfer function design because the opacity parameters change in the course of transfer function exploration.

Kohlmann et al [KBEGK07] also presented a similar system for medical visualization. In this system, the goal is to show a 3D view of user selected feature. Users choose a point in the 2D slice view panel and the system finds the best viewpoint for the point. The necessary information to determine the best viewpoint is precomputed and saved as a deformed sphere where a large radius denotes a good viewpoint. This

approach also combines multiple parameters, such as the axis os patient orientation, viewpoint history, the orientation of local shape, and visibility. The visibility is computed for more than 600 different directions. Then, from the combined information, the system computes the best viewpoint for the selected area of interest.

Tao et al. [TLB+09] extended the work of Ji and Shen. In this work, the best viewpoint renders volume data showing both global structure and local details. Previous approaches only focused on global features, but Tao et al. put emphasis on the local features. The local features can be found by comparing two volumes, original volume and filtered volume. The filtered volume is created by bilateral filtering. The filtering preserves the boundary structure of the data while local details in the data are smoothed out. Then the difference between the filtered volume and the original volume captures the local details in the volume. This idea is useful if one side of a die contains small patterns. If a viewpoint selection algorithm focuses on the global structure of the die, the algorithm will ignore the pattern. However, because Tao et al. capture the local details, the algorithm can successfully balance between the optimal viewpoints for global structures and for local details. The computational aspect of this, compared to previous approaches, is that one should compute this bilateral filter image, which is known to be expensive [PKTD09]. Moreover, the algorithm uses gradients of both original and filtered volume, which adds up to a greater computational workload.

Table 6.1 summarizes 1) how each of the representative viewpoint selection algorithms defines "best view," 2) on which element they defined the probability distribution for entropy function, and 3) what they measure with the probability distribution. Note that Lee et al. [LVJ05] do not use the term entropy. The mesh saliency feature values are defined on each vertex in the 3D mesh, and the best viewpoint is the one that the most visibly shows salient features. This is very similar to other approaches. Table 6.1 re-confirms that there is no global consensus about what a good viewpoint is. However, the common factor among them is that all the approaches find interesting objects in the data and compute a viewpoint that can show the objects.

Table 6.1: The definitions of "good" viewpoint are different among previous approaches. Moreover, the features for which each algorithm searches and the domain on which the feature is defined are also distinct.

Approach	Definition of Best View	Entropy Domain	Entropy Measure
Vázquez et al. [VFSH01]	A viewpoint that provides a high amount of information: more mesh faces visible and minimum number of degenerative faces.	Mesh face in 3D mesh	Projected area of mesh faces
Takahashi et al. [TFTN05]	A viewpoint that arranges the feature components effectively.	Isosurfaces extracted from 3D volume	Projected area of isosurface faces
Bordoloi and Shen [BS05]	A viewpoint that renders important voxels highly visible.	Voxels in 3D volume	Visibility, opacity, and opacity frequency
Lee et al. [LVJ05]	A viewpoint where salient features are the most visible.	Vertices in 3D mesh	Mesh saliency
Ji and Shen [JS06]	A viewpoint where three features, high opacity, distinct colors, and high curvature, are highly visible.	Pixels in rendered images	Opacity, color, and curvature
Mühler et al. [MNTP07]	A viewpoint that maximizes the weighted sum of various parameters.	Pixels in rendered images	opacity, occlusion effects, preference, view stability
Tao et al. [TLB+09]	A balanced viewpoint between overall orientation of features and visible details on boundary structure.	Pixels in rendered images	Global structure and local details

6.3 Interaction with Transfer Function

As discussed in Chapter 4 and Chapter 5, in volume rendering, finding a good transfer function for effective color and opacity is a very important step. During this transfer function exploration, viewpoint selection algorithms assist users in understanding the current setting of transfer function. Figure 6.2 illustrates how the two components interact with each other to help users better understand the data and quickly obtain the optimal visual appearances. Users explore the transfer function domain to tune the appearance so that interesting objects in the data stand out in color and opacity. This process is usually done by trial-and-error, and it takes time to find a good setting. Whenever users set a transfer function, they have to visually check whether their settings are what they are looking for. Because the data we visualize is 3D objects, there is occlusion among the objects, and thus users have to rotate the scene to find a viewpoint that has minimal occlusion among the objects in which they are interested. Viewpoint selection algorithms can greatly shorten the process of finding a good viewpoint. The viewpoint chosen by the viewpoint selection algorithm can provide 1) overall appearance and 2) minimal occlusion among interesting points. If a viewpoint selection algorithm can achieve the goal successfully, users can understand the scene and whether the current setting of transfer function is correct. The two components continue interacting with each other until users find a good visual appearance.

This interaction has an a bigger influence on the exploration for multi-channel data. As discussed in Chapter 2, each channel shows a completely different part of the specimen. Therefore, the rendered image of each channel varies greatly depending on which channels are on or off. With the advanced transfer function for multi-channel data introduced in Chapter 5, the points of interest are located in a different place in the 3D space. Moreover, the appearance of each point of interest varies significantly. Therefore, the viewpoint selection process can assist users in understanding scenes and checking transfer function settings.

However, there is one critical requirement for viewpoint selection algorithms. Due to the advanced graphics hardware, volume rendering, as well as transfer function exploration runs in real-time [KKH02]. Therefore, in order for the interaction described in Figure 6.2 to happen in real-time, each component must run as quickly as possible.

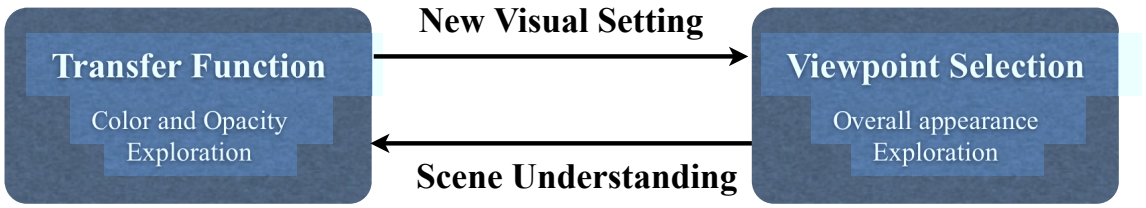


Figure 6.2: Interaction betweeen transfer function and viewpoint selection algorithm. A new setting of transfer function requires the viewpoint selection component to find a different viewpoint that best describes the new setting, allowing users to understand the current scene.

Transfer function update is done by updating the look-up table mapping from the original intensity to opacity and color. This look-up table is significantly smaller than the original intensity values, and the data transfer bandwidth between CPU and GPU is 16GB/s with PCI-Express 2.0. Therefore, if the remaining component, viewpoint selection, can run in real-time or at least fast enough, users can explore the volume with the continuing interaction between transfer function and viewpoint selection.

While previous approaches have shown promising results over various data sets, the major limitation of information theoretic frameworks is the fact that they rely on an exhaustive search over a large set of samples. This method has to trade off between the accuracy of the solution and the computation time. For example, suppose we take 400 uniform samples on the viewing sphere. This is, on average, 20 samples around one circumference. Then, the granularity of the sample is $\pi/10$. If the real solution is on a small peak not captured by the granularity, increasing the number of samples is the only way to correctly locate the solution. The execution time then increases in proportion to the sample size because the framework requires rendering for each sample. Suppose the rendering can be done at a rate of 50 frames per second. Even if we exclude feature computation time and the time for reading back the rendered image from GPU, processing takes at least 8 seconds for 400 samples. This limitation of information theoretic frameworks significantly hinders users from smooth interaction between transfer function and viewpoint selection.

The performance aspect of exhaustive search algorithms has rarely been studied in previous viewpoint selection frameworks except by Vázquez and Sbert [VS03] and Lee et al. [LVJ05]. Vázquez and Sbert estimate entropy values of new positions by using

the information on visibility of faces and already computed entropy of neighbor points. The algorithm starts from a set of initial points and computes the entropy for the points. Then, it estimates the entropy values for their middle points. If the estimate for a point is greater than correctly computed entropy values, the algorithm iterates the estimation and computation of the entropy around the point.

Lee et al. [LVJ05] reduced the search time by employing a gradient-descent heuristic. The algorithm starts at a random point and expands toward its neighbor points. Gradient values are computed for the neighbor points, and the algorithm picks one that has the largest positive gradient.

Both algorithms certainly narrow down the search space, but they are heuristic algorithms, and there is a possiblity that the algorithms miss the globally optimal solution.

6.4 Summary

In this Chapter, we have defined the viewpoint selection problem and reviewed previous approaches. Most approaches propose their own view descriptor, measuring the amount of important features from each viewpoint. Numerous view descriptors indicate that there is no concensus on what the best view of 3D data is, but the common goal is to help users understand their data effectively.

The view-dependent descriptors require searching over a set of candidate viewpoints, which often takes too long to be used as a real-time viewpoint suggestion. This limitation is critical in transfer function exploration. Users would like to understand the volume with the newly set transfer function, and the viewpoint selection algorithm is an essential component in the transfer function exploration.

Acknowledgements

This chapter, in part, is based on material that is in preparation for submission: "Real-Time Data-Centric Viewpoint Selection" by Han Suk Kim, Didem Unat, Scott B. Baden, and Jürgen P. Schulze. The dissertation author was the primary investigator and

author of this paper.

Chapter 7

Real-Time Data-Centric Viewpoint Selection

In this chapter, we propose a real-time viewpoint selection algorithm to solve the problem discussed in the previous chapter. For this goal, we first define a good viewpoint in a different way from the information theoretic approaches. When viewers investigate a volume data set, they mostly focus on interesting parts of the data. This has been a driving force for transfer function technologies to develop many ways to highlight interesting parts of the data. We also employ the notion of "interesting parts" into our algorithm. The best view in our algorithm is a scene where all the interesting parts of a volume data set are shown with the smallest possible amount of occlusion. With this definition, we have to answer three questions: 1) what are interesting parts?; 2) how can we find the view that has the least amount of occlusion among interesting features?; and 3) how do we compute the view direction efficiently?

We solve the first problem with an algorithm from computer vision. We extend the Harris interest point detection algorithm [HS88] to find interesting features in 3D volume data sets. The algorithm selects corners of an object and/or areas that have high intensity compared to neighboring voxels as interest points. For the second problem, we formulate the problem as an optimization problem and the objective of the optimization is maximizing the variance of projected feature points. If all the interesting features are laid out without any overlapping with each other, viewers have a better chance to understand the data quickly. One way of stretching out a set of points in a plane is

to maximize the variance of them. By finding a plane on which the variance of feature points is maximized, we can minimize the overlap between feature points. This optimization is formulated as a Pricipal Component Analysis problem.

Our algorithm runs in real-time because of two main strengths: the simplicity of our algorithm, and an efficient implementation on the graphics processing unit (GPU). These two strenghths address the third problem regarding the efficiency of our approach. Our algorithm computes one feature value for each point and the optimization problem takes only a small set of points as an input. However, other approaches compute a set of features and perform volume rendering many, often hundreds of, times. We further reduce the computation cost through hardware acceleration implementing the critical parts of our algorithm on a GPU with many stream processors.

Having a real-time viewpoint selection algorithm is important because it can greatly help volume rendering users understand the volume data sets throughout the transfer function exploration. Nowadays, volume rendering is very interactive and the transfer function in volume rendering dramatically changes the appearance of the volume [KKH01]. Once our viewpoint selection algorithm is integrated into the transfer function, users can immediately see whether or not the new settings of the transfer function produce good images without any further exploration.

The remainder of this chapter is organized as follows: In Section 7.1, we introduce our feature selection algorithm, Section 7.2 describes the details about how we compute the best view direction from the features. Section 7.3 discusses the issues in implementing and optimizing our algorithm to run in real-time. Section 7.4 presents performance results for several typical data sets. We also discuss possible applications and some limitations of our real-time viewpoint selection algorithm in Section 7.6. Finally, we conclude this paper and suggest future work in Section 7.7.

7.1 Feature Selection

The goal of our feature selection algorithm is to find all the features in a data set that would help the viewer understand the object. For example, corners of an object and high intensity points are the important features that can help the viewer understand

the object. In order to find the features, our algorithm has two steps, *feature selection* and *filtering*. The feature selection algorithm computes a score for each voxel and the filtering picks up only a small set of voxels, *interest points*.

7.1.1 Feature Selection Algorithm

Our feature selection algorithm is based on the Harris interest point detection algorithm [HS88]. We first briefly review the Harris interest point detection algorithm and then present how we extend it for our problem.

Background

The Harris interest point detection algorithm [HS88] produces a score for each pixel in 2D images. Positive score values correspond to interest points in the image. A pixel is selected as a point of interest if there is a large change both in the X-axis and Y-axis. For example, the algorithm gives a large positive score to a corner of a shape or a point that has a high intensity. On the other hand, the algorithm maps points in a flat area to zero and edges of an object to negative values. Corner points are important features for capturing the geometry of an object. If all the corners of an object are exposed when viewed from a viewpoint, users can understand the overall geometry of the object. High opacity points, which are also captured as the Harris interest points, are also crucial features, because users usually set the opacity of important areas high so that they can see the objects clearly.

The algorithm measures the change $E(x,y)$ in intensity at pixel (x,y):

$$E(x,y) = \Sigma_{u,v} g(u,v) |I(x+u, y+v) - I(x,y)|^2 \tag{7.1}$$

where $I(x,y)$ is the opacity value at image coordinate (x,y) and $g(u,v)$ is defined as a Gaussian weight around (x,y). If a point (x,y) is in an area that has less change in intensity, $E(x,y)$ will be close to 0. On the other hand, around the area that has a large change, $E(x,y)$ is also large.

The computation of $E(x,y)$ is done using the Taylor expansion:

$$\begin{aligned} E(x,y) &= \Sigma_{u,v} g(u,v) |I(x,y) + xI_x(x,y) + yI_y(x,y) - I(x,y)|^2 \\ &= \Sigma_{u,v} g(u,v) \left[xI_x(x,y) + yI_y(x,y)\right]^2 \\ &= Ax^2 + 2Cxy + By^2 \end{aligned} \tag{7.2}$$

where A, B, and C are a Gaussian convolution of I_x^2, I_y^2, and $I_x I_y$, respectively:

$$\begin{aligned} A &= g \otimes I_x^2 \\ B &= g \otimes I_y^2 \\ C &= g \otimes (I_x I_y) \end{aligned}$$

Then, $E(x,y)$ can be written in matrix form:

$$E = \begin{bmatrix} x & y \end{bmatrix} \begin{bmatrix} A & C \\ C & B \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} \tag{7.3}$$

Therefore, the intensity change in E is characterized by the 2x2 matrix in Equation 7.3, which we refer as M, and if the change is large in both directions, the two eigenvalues of M should be large. The eigenvalues of M can be computed by singular value decomposition (SVD) but it needs a long computation. In order to avoid the SVD computation to find eigenvalues for each point, Harris and Stephens [HS88] proposed a response function as follows:

$$R = det(M) - kTrace(M) \tag{7.4}$$

where k is an empirical constant, usually set between 0.04 and 0.06 for 2D images. Points that have large changes in both directions have two large eigenvalues of M. If the eigenvalues are significantly large, $det(M)$ impacts more on the value of R than $Trace(M)$. If only one eigenvalue is significant, R becomes negative as $Trace(M)$ is greater in the equation. In this case, the points represent edges because the change is significant in only one direction. We refer to R as "Harris score" hereafter.

Extension to 3D

In direct volume visualization, data is stored on a regular 3D grid. Thus, in order to use our approach with such data, we need to extend the algorithm to the 3D spatial

domain. Previously, Laptev and Lindeberg [LL03] extended the algorithm to the spatio-temporal domain, and Sipiran and Bustos [SB10] explored how the algorithm can be extended for 3D mesh data. We adopt the same structure as in Laptev and Lindeberg, but, in our case, the third domain is also a spatial domain. The change function $E(x,y,z)$ is defined as:

$$E(x,y,z) = \Sigma_{u,v,w} g(u,v,w)|I(x+u,y+v,z+w) - I(x,y,z)|^2 \tag{7.5}$$

and the Taylor expansion for $I(x+u,y+v,z+w)$ is approximated as follows:

$$I(x+u,y+v,z+w) \approx I(x,y,z) + xI_x + yI_y + zI_z + O(x^2,y^2,z^2) \tag{7.6}$$

Following the same logic as in Equation 7.2 yields the matrix M to be defined as follows:

$$M = g \otimes \begin{bmatrix} I_x^2 & I_xI_y & I_xI_z \\ I_xI_y & I_y^2 & I_yI_z \\ I_xI_z & I_yI_z & I_z^2 \end{bmatrix} \tag{7.7}$$

This involves 3D convolution for each voxel, which is computationally expensive. By using GPU acceleration techniques, we were able to significantly improve the performance of this algorithm. We further discuss the performance of this algorithm in Section 7.3.

In our algorithm, the width of the Gaussian convolution kernel is set to 5; the convolution is a weighted sum of $5 \times 5 \times 5$ patch around a point (x,y,z). The variance of the Gaussian kernel is set to 1.5. The sensitivity constant k is set to 0.004.

Once M is computed for each point, the Harris score can be computed with Equation 7.4 but the number of arithmetic operations needed to get the score is larger than the original 2D algorithm.

7.1.2 Filtering

The extended Harris interest point detection algorithm produces a score for each voxel. The large score points are either at corners of the geometry or in bright areas.

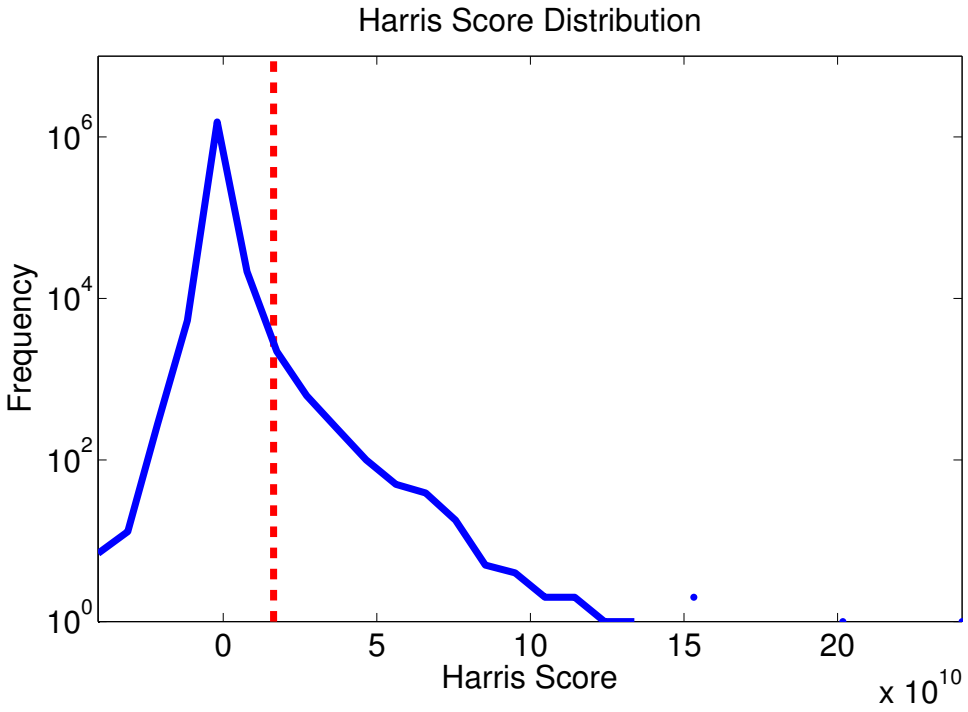


Figure 7.1: Harris score distribution of the Tooth data, which is used in both Ji and Shen [JS06] and Tao et al. [TLB+09]. The solid blue line shows the histogram of the Harris score of the Tooth data set. Positive values are considered interest points or corner points, near zero points are flat areas, and negative values indicate edges. The dotted red line shows the threshold for X. All the points on the right side of the red line are added to X.

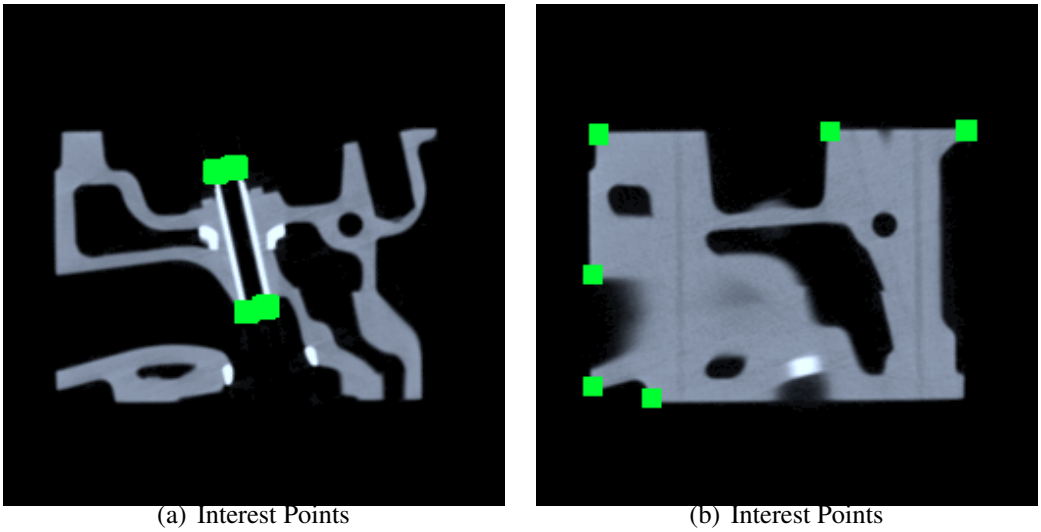
Because we are only interested in the largest score points, we need to filter out points that represent flat regions or edges. The algorithm for selecting interest points scans the entire score data, picking the largest N values. While we scan the score values, we store the largest N values in a priority queue, and whenever we see a new value larger than the smallest of the queue, we replace it with the new value. This algorithm runs in linear time asymptotically, although it requires many operations in the constant-size priority queue.

The set of large score points can be reorganized as a matrix $X \in \mathbb{R}^{N\times 3}$, where each row represents the coordinate of the points selected as interest points in the previous step.

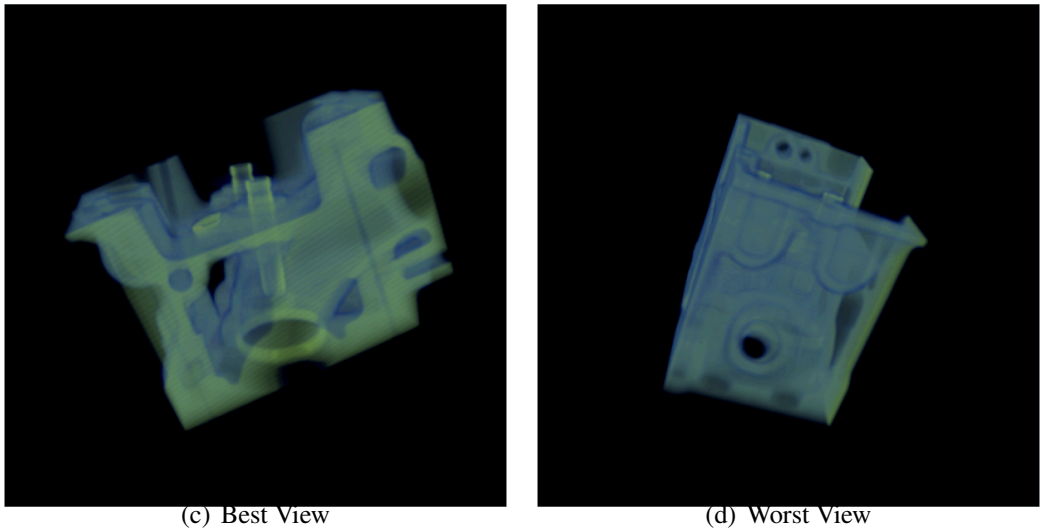
$$X = \begin{bmatrix} x_1 & y_1 & z_1 \\ & \vdots & \\ x_N & y_N & z_N \end{bmatrix} \tag{7.8}$$

Figure 7.1 shows the Harris score distribution of the Tooth data set. The maximum Harris score for this data set is 2.5×10^{11} and the minimum is -4.6×10^{10}. The distribution has a peak around 0, which is typical for many volume data sets because empty regions have scores around 0. Note that, however, having both positive and negative Harris scores is not typical. Depending on the characteristics of the data, it may not have many corner points, for instance, when an object has a very smooth surface. The constant k plays an important role in determining the shape of distribution.

Figure 7.1 also shows the threshold for the largest N values. All the points that have a Harris score greater than the threshold are included in X, and there are 2048 such points in this example. The position of the threshold shows that if we include more points, the line moves to the left, which will include some points that may not be interest points. The number of elements N in X is empirically determined by checking the distributions of R. If we select too few points, we fail to capture all the important points in the data set. On the other hand, if we include too many points, we may end up adding noisy data points, for instance, points in flat areas or at edges. In addition, having many points in X increases the computational cost 1) in the filtering because the size of the priority queue increases, and 2) in the optimization solver discussed in Section 7.2 because X impacts the performance of the solver.



(a) Interest Points (b) Interest Points



(c) Best View (d) Worst View

Figure 7.2: Results for the Engine Block CT Scan data from General Electric, USA. Figure 7.2(a) and Figure 7.2(b) show the result of the 3D Harris interest point detection. We highlighted interest points with green squares. In Figure 7.2(a), four corners of the cylinder are selected as interest points. Figure 7.2(b) shows the outermost slice of the engine block and 6 corners are detected as interest corners. The best view in Figure 7.2(c) renders the volume from the side, but note that the direction is slightly tilted to expose the top of the cylinders. The worst view renders the volume from the side. This view does not contain as much information as Figure 7.2(c), preventing viewers from understanding the data.

Figure 7.2(a) and Figure 7.2(b) show examples of interest points. We applied the interest point detection algorithm and selected only large score points as described in this section. Each green rectangle denotes an interest point detected by our algorithm. The Engine Block CT Scan data set has two cylinders in the block, and Figure 7.2(a) shows that the top and bottom of the cylinders are selected as interest points. Figure 7.2(b) shows that the algorithm successfully identifies most corners in the last slice for the block. If we add more points in X by finding more large values, additional corners will be eventually added, but we did not see any improvement in the viewpoint selection.

7.2 View Selection

The previous steps generate a set of feature points. Those points are clustered around the corners of high opacity region. Those points are important in viewpoint selection 1) because we do not want the corners of an object to overlap each other and 2) because we want to avoid the salient features, which users highlights with transfer function manipulation, to occlude each other. Corner points capture the geometry of the objects in the data. If the corner points have minimum occlusion, there is a higher chance that users can see the data from the canonical view of the data, revealing more details of the data. If the data has outstanding features, which are captured in the Harris interest points, and if those features has minimum occlusion, we can still see important objects in the data. Because the Harris interest point algorithm favors high opacity values, high intensity points, i.e., opaque objects, are more likely to be selected as the Harris interest points. If our view selection algorithm can minimize the occlusion between the points, the occlusion between opaque objects in the data is also minimized.

Therefore, in the next step of finding the best viewpoint, we need to minimize the occlusion between the feature points. Minimizing the occlusion can be thought as maximizing the distance between the feature points, i.e. spreading out far apart from each other. The direction that minimizing the occlusion can be computed by solving an optimization problem, which has the same formulation as Principal Component Analysis (PCA) [Pea01].

7.2.1 Principal Component Analysis

We employ PCA to find the best view direction from the matrix X. The main idea of PCA is to find the direction $\hat{u}$ that maximizes the variance of the projected points of X when projected to $\hat{u}$. The optimization formulation is:

$$\begin{aligned} &\text{maximize } Var(uX) \\ &\text{subject to } ||\vec{u}||^2 = 1 \end{aligned} \tag{7.9}$$

The detail of how to find the solution for the optimization problem in Equation 7.9 can be found in Appendix A. but the solution is the eigenvector with the largest eigenvalue of the covariance matrix of X. That is, the best viewpoint is the solution of $X^T X u = \lambda u$. Note also that the solution is the global maximum and therefore it always gives the best possible direction out of all possible choices of u. This means that our algorithm does not trade accuracy for performance. In addition, there is no trial-and-error parameter tuning process to find the solution.

By maximizing the variance of projected points, we expect that we minimize the overlap between the selected interest points. By placing interest points far apart, we can see all the interest points in the data set with little — or ideally, no — occlusion. Therefore, our best viewpoint guarantees that one can see as many details of the volume as possible, although we project 3D data onto a 2D plane.

We use SVD [FP02] for the actual computation. SVD decomposes the input matrix X into $U\Sigma V^T$ and the right singular vector V's are the eigenvectors of $X^T X$. Suppose SVD produced V as $[v_1|v_2|v_3]$, then the interest points yield the maximum variance along direction v_1. In order to see the largest variance in the rendered image, the camera's view direction should be v_3, looking at the center of the volume, because v_3 is orthogonal to two directions v_1 and v_2, the two axes in which the points achieve maximum variance. On the other hand, if the points are projected along direction v_3, the variance of the projected points are the minimum among all possible choices of u in Equation 7.9. Thus, the "worst viewing" direction is from v_1, orthogonal to v_3 and v_2.

We also considered using the Harris score values as weights for uX in Equation 7.9 so that larger weight points should be placed far apart even if small weight

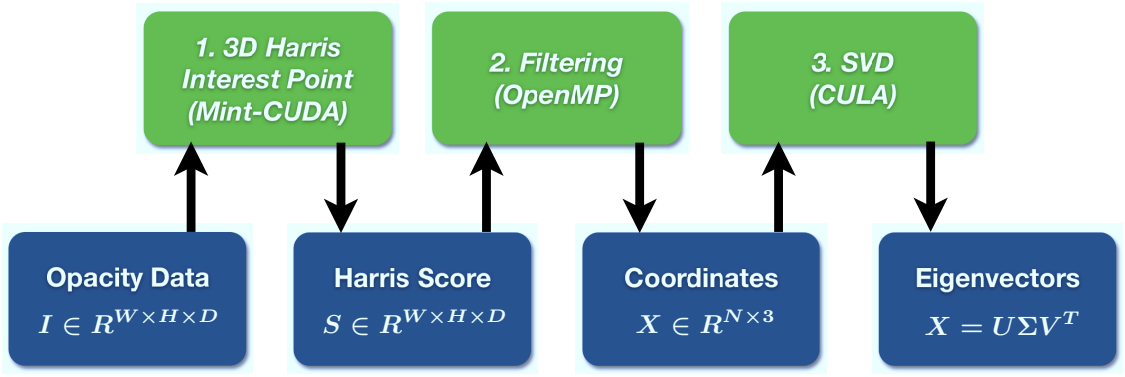


Figure 7.3: The computational pipeline for our viewpoint selection algorithm. The input to the algorithm is 3D opacity data stored in a structured grid. The value ranges from 0 to 255 (8 bits). The 3D Harris interest point detection algorithm produces the Harris score values for all voxels. The filtering step searches the largest values of the score and the (x, y, z) coordinates for the values. Finally, the last step executes the SVD of X and computes the eigenvectors of X. Note that the "3D Harris Interesting Point" and "SVD Algorithm" steps run on the GPU, whereas the filtering step is done on a multi-core CPU with OpenMP.

points are close to each other. This is because if two points which have large weights and are placed close to one another, the variance is penalized heavily. However, the Harris score values do not proportionally correspond to the likelihood of being interest points, i.e., a point having twice as large a score as another does not mean the point is twice more likely to be an interest point. Moreover, the percentage of points we select is relatively small and those points are all important features in the data. Thus, we do not leverage the Harris score values themselves in the optimization.

7.3 Implementation

We implemented the algorithm described in Section 7.1 and Section 7.2. There are three main kernels to compute the best viewpoint: 1) 3D Harris Corner Point, 2) Filtering, and 3) PCA. Figure 7.3 shows the pipeline of the algorithm. The input to the algorithm is the opacity data modified by the transfer function. The output of the algorithm is the best direction for displaying the volume given the settings of the transfer function.

The 3D Harris interest point algorithm (Section 7.1.1) takes the opacity data and

produces a Harris score for each voxel. This kernel is the most computationally challenging part of the three kernels, as it requires convolution, which performs many memory accesses and arithmetic operations per voxel. Since our goal is to make the entire algorithm run in real-time, an implementation on a traditional multi-core architecture would not have met our goal. However, the algorithm is a good candidate for GPU acceleration because it is highly data parallel and can benefit from the high bandwidth to GPU memory. Therefore, we ported the algorithm to the GPU. We use the Mint source-to-source translator [UCB11] to automatically generate the CUDA (Compute Unified Device Architecture [NBGS08]) code for the 3D Harris interest point algorithm. The Mint translator takes C code with simple annotations and generates optimized CUDA code. The automatic optimizations in the translator include shared memory optimization to reduce memory access operations, boundary condition optimization, loop unrolling, and constant folding. The Mint-generated CUDA implementation allowed us to achieve a substantial performance improvement over the single CPU implementation, delivering a 45x speedup. More details about the parallelization and the implementation on GPU are described in Unat et al. [UKSB11].

The second kernel is filtering (Section 7.1.2). The input to this kernel is the Harris score computed in the previous step and the output is a matrix X defined in Equation 7.8, i.e., the list of coordinates for largest Harris scores. Because the Harris score is stored in GPU memory and we compute the filtering step on the CPU, we perform memory transfer from GPU memory to CPU memory. The asymptotical running time of this step is linear in the number of voxels. However, the cost of adding and removing elements in the priority queue (the asymptotical running time for these operations is $O(\log N)$, where N is the size of the queue) was not trivial. In order to improve the performance, we parallelized this kernel with OpenMP [DM98]. The parallelization algorithm works as follows: multiple threads equally divide the entire Harris score data and each thread searches the largest N values in its own priority queue. After the parallel section is done, a single thread merges the multiple queues into one. We set N to 2048 for all our results.

The last step is to compute the eigenvectors of X. The output of this step is a vector in 3D, which is the best viewpoint determined by our algorithm. Because the

size of X is a constant, 2048×3 in our case, the computational overhead for this step is trivial compared to the previous steps. We used the CULA library [Pho11], a linear algebra package in CUDA. Thus, the only part not implemented in CUDA is the second kernel: filtering. We are also in the process of off-loading the second kernel to the GPU, although the cost of the memory transfers between CPU and GPU memory for the kernel is not as high.

7.4 Experiment

First, we discuss the running time of our algorithm to show the algorithm runs efficiently enough to be used as part of a real-time volume rendering system. The quality of the algorithm is also discussed by showing rendered images of several popular data sets.

7.4.1 Performance

All performance results presented in this chapter were obtained on a workstation comprising a quad-core Intel Xeon CPU (2.0GHz) with 16GB of main memory and 4 Tesla C1060 GPUs. Our implementation currently uses only one of the GPUs. As mentioned in Section 7.3, we used the Mint translator, version 0.3, to generate the CUDA code. This code was subsequently compiled with the NVIDIA nvcc compiler and linked with the CULA R10 linbrary. We used version 3.2 of the CUDA Toolkit. The operating system for the machine is Ubuntu 10.04.2 and the GCC version is 4.4.3.

We measured the running time 10 times for each case and report the minimum in Table 7.1. We measured three major components of the algorithm separately: 1) 3D Harris interest point detection, 2) filtering, and 3) principal component analysis. The column for "Running Time" includes all three components as well as memory transfer time between main memory and GPU memory. We excluded the initialization code that allocates memory and reads opacity data from files, which runs only once in volume rendering systems.

Unlike other information theoretic frameworks, our algorithm does not include any rendering step. Thus, the running time only depends on the data size. More specifi-

Table 7.1: Performance numbers for various volume data sets. The last three columns show the running times of each step: 1) computing the 3D Harris interest point detection, 2) selecting the largest 2048 values and coordinates from the Harris score, 3) computing eigenvectors to find the best view direction. The Running Time column includes memory transfer time as well as the time for the three steps.

Data Set	Size	Number of Voxels	Running Time	Harris	Filtering	PCA
CT-Knee	$379 \times 229 \times 305$	26,471,255	**0.8450s**	0.797s (94.26%)	0.043s (0.51%)	5.74ms (0.07%)
Foot	$256 \times 256 \times 256$	16,777,216	**0.7331s**	0.340s (53.17%)	0.231s (31.52%)	5.23ms (0.71%)
Engine	$256 \times 256 \times 256$	16,777,216	**0.5369s**	0.340s (72.61%)	0.035s (6.52%)	5.23ms (0.97%)
Lobster	$301 \times 324 \times 56$	5,461,344	**0.2298s**	0.132s (57.61%)	0.045s (19.58%)	5.05ms (2.20%)
Orange	$256 \times 256 \times 64$	4,194,304	**0.2574s**	0.099s (38.31%)	0.122s (47.41%)	5.09ms (1.98%)
Tooth	$94 \times 103 \times 161$	1,558,802	**0.0838s**	0.044s (52.29%)	0.018s (21.49%)	5.06ms (6.04%)
Cross	$66 \times 66 \times 66$	287,496	**0.0221s**	0.011s (47.92%)	0.003s (13.56%)	5.30ms (23.96%)
Box	$64 \times 64 \times 64$	262,144	**0.0218s**	0.010s (44.56%)	0.002s (9.19%)	5.16ms (23.70%)

Table 7.2: Performance comparison of the Harris interest point detection algorithm among different architectures. The serial version is ran on a single CPU, the OpenMP version is ran on a quad-core, and the Mint-CUDA version is implemented in Mint-generated CUDA. The first three columns show the running time of the Harris interest point detection algorithm in seconds. The last two columns show the speedup of Mint-CUDA against the serial version and the OpenMP version. The CUDA version achieved huge speedups ranging between 24 to 45 against the serial version. Even compared with the OpenMP version, the CUDA version ran more than 6 times faster.

Data Set	**Running Time**			**Speedup**	
	Serial	**OpenMP**	**Mint-CUDA**	**Serial**	**OpenMP**
CT-Knee	24.474s	6.392s	0.797s	30.71	7.58
Foot	15.580s	3.933s	0.340s	45.82	11.57
Engine	15.375s	3.927s	0.340s	45.22	11.55
Lobster	5.012s	1.466s	0.132s	37.97	11.11
Orange	3.882s	0.986s	0.099s	39.21	9.96
Tooth	1.427s	0.426s	0.044s	32.43	9.68
Cross	0.264s	0.072s	0.0106s	24.91	6.79
Box	0.244s	0.062s	0.010s	24.40	6.20

cally, the 3D Harris interest point detection algorithm runs in linear time in terms of the number of voxels in the data. Table 7.1 confirms the linearity.

The filtering step is also proportional to the size of data. However, another important factor for this is the distribution of the Harris scores. If the scores are sorted in an increasing order, the algorithm constantly removes the smallest element from the priority queue and pushes the new element into the queue. Although the queue size is not as big as the data size, we found that it could incur a significant overhead. The final step, PCA, runs over a constant size matrix X regardless of the volume data set size. There was a small variation between the data sets, which might come from measurement variations or varying convergence speed in the SVD algorithm.

Of the three computational components, the 3D Harris interest point detection is the most compute-intensive kernel. This is due to the fact that it computes value per voxel. However, we were able to reduce the running time with Mint and CUDA. Ta-

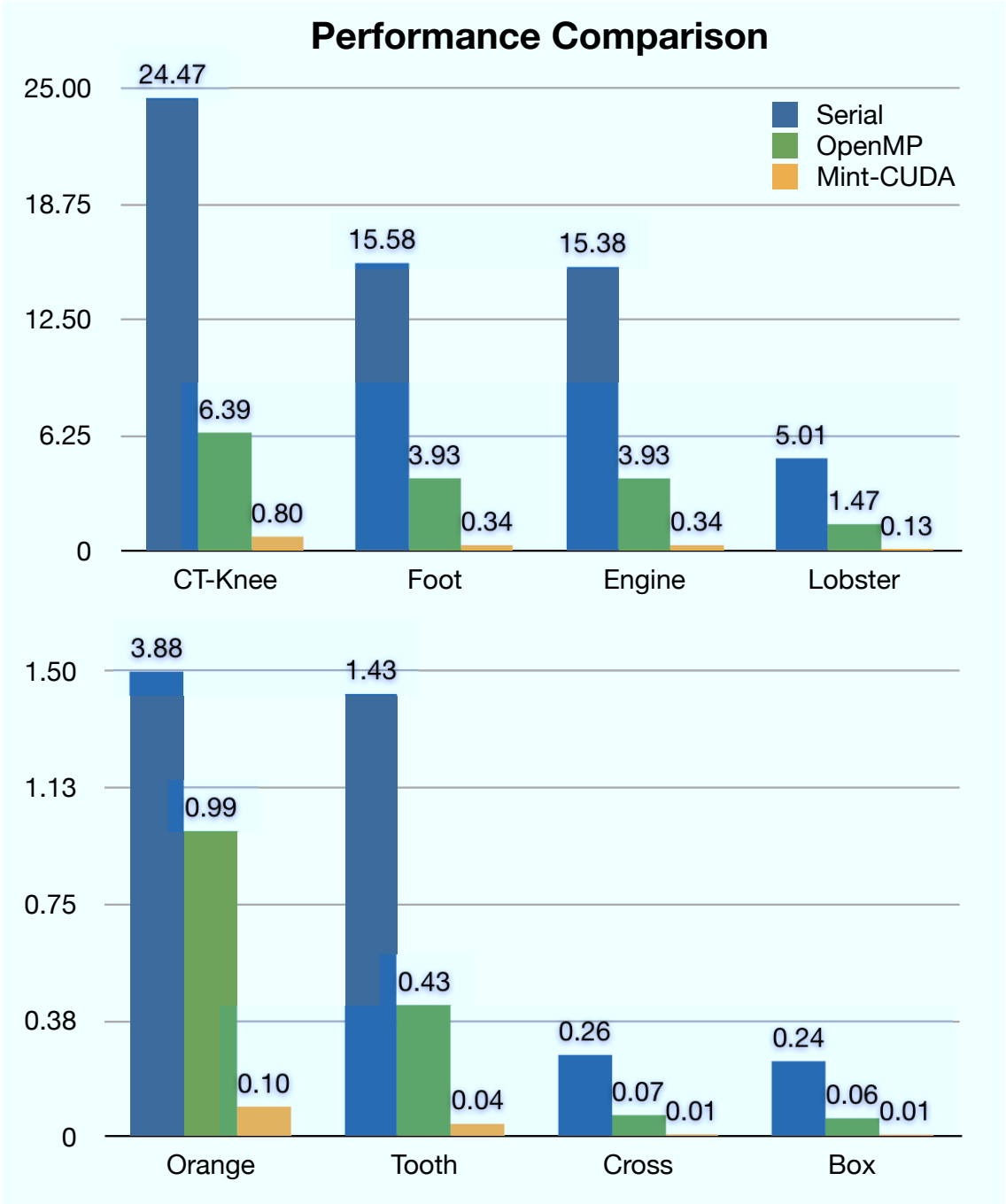


Figure 7.4: Performance comparison among three different methods: single CPU, multi-core with OpenMP, and CUDA with Mint. CUDA implementation based on the Mint programming model shows huge speedups ranging between 24 to 45 against the single CPU implementation and about 6 times speedups against the multi-threaded implementation with OpenMP.

ble 7.2 and Figure 7.4 report the speedup of the Harris interest point detection algorithm among our three implementations in three different methods: singe CPU, multi-core with OpenMP, and CUDA. The running time column shows the running time measured for the Harris interest point detection algorithm. The serial version computes the values with a single CPU, whereas the OpenMP version utilizes all 4 cores available. The Mint generated CUDA version uses one of the available GPUs for the main computation. The last two columns, Speedup, show the speedup of the Mint-CUDA version against the serial and OpenMP version. The largest speedup achieved is with the Foot and Engine data, 45x speedup against the serial version and 11x speedup against the OpenMP version. The benefit decreases as the size of the volume data set decreases. The overhead of transfering the data back and forth between the GPU memory and the main memory and the latency for launching the computational kernel on GPU caused the decrease. However, even with the smallest data set, Box, the speedup and the running time itself indicates we achieved a substantial performance improvement against other implementations.

7.4.2 Result for Multi-Channel Data

We applied our method to a multi-channel data set. We use the zebrafish head data, courtesy of Hideo Otsuna and Yong Wan from the University of Utah. The zebrafish head data consists of three channels: the first channel, rendered in red, shows eye muscles; the second channel, rendered in green, expresses eye, retina ganglion cells and tectum; and the third channel, rendered in blue, expresses cell nuclei.

All the rendered images in this chapter are rendered with a 3D texture-based rendering algorithm with alpha blending. The viewport size is set to 512×512. The center of each volume is located at the origin and the view direction is set to look at the origin.

Figure 7.5 and Figure 7.6 show the overall screenshots showing different organs in the data. The figure shows the eye and brain region. The tectum and cells in the eye are colored in green. The eye muscles wraps around the eye region. The blue channel denoting cell nuclei is wraps around the organs. Because scientists are usually more interested in channel 1 and 2, Figure 7.5 shows only two channels turned on.

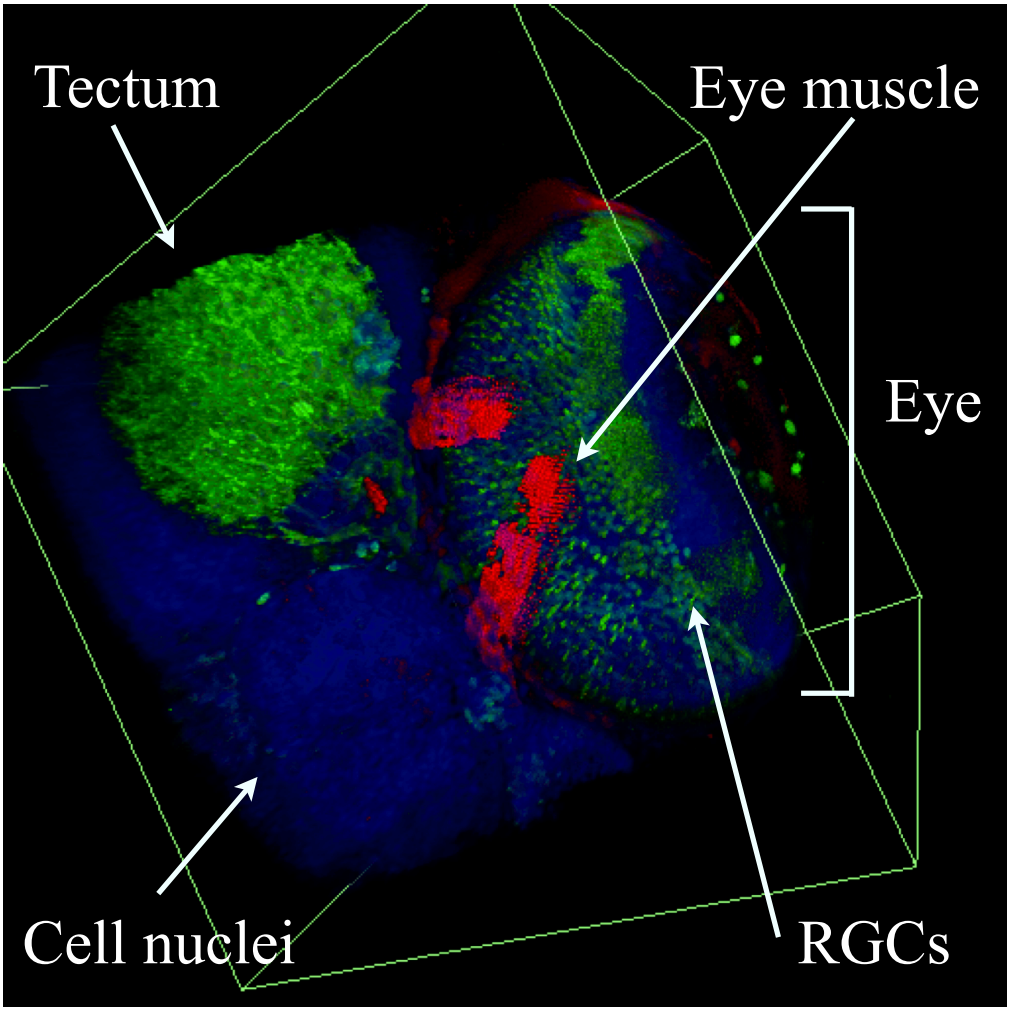


Figure 7.5: Rendered images of zebrafish head with all three channels turned on. The data show the area around the eye and the tectum. The eye muscle colored in red wraps around the retina ganglion cells (RGCs).

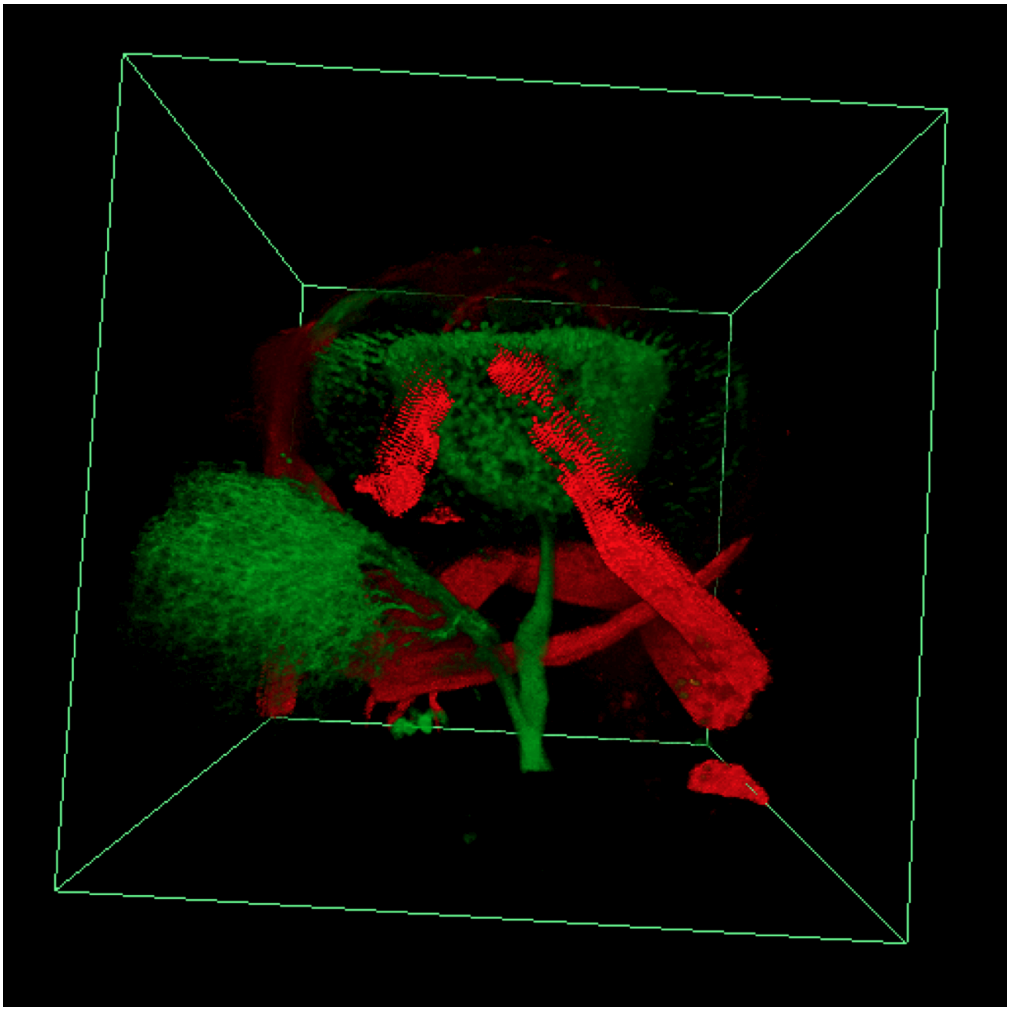


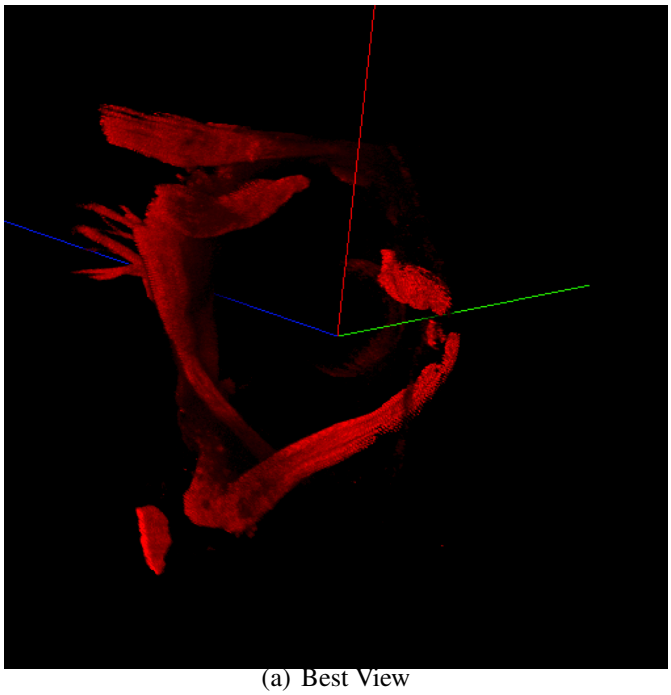
Figure 7.6: Rendered images of zebrafish head with only two channels turned on. With the blue channel off, one can clearly see where the eye muscles are located around the RGCs.

As each channel already shows distinct objects, we show results of our view selection algorithm for each channel. Figure 7.7 shows the result for channel 1. From Figure 7.7(a), one can clearly understand how the eye muscles are spread out in the specimen. Figure 7.7(b), which is the worst viewpoint from our algorithm, prevents viewers from understanding the overall structure of objects expressed in channel 1. Figure 7.8 shows the second channel. Figure 7.8(a) clearly shows the two components, tectum and RGCs, in the data, with no occlusion, whereas in Figure 7.8(b), all the objects overlap each other . Figure 7.9 shows the third channel. The third channel contains cell nuclei, the small details of which can be seen in Figure 7.9(a). However, a side view, as shown in Figure 7.9(b), fails to display those details.

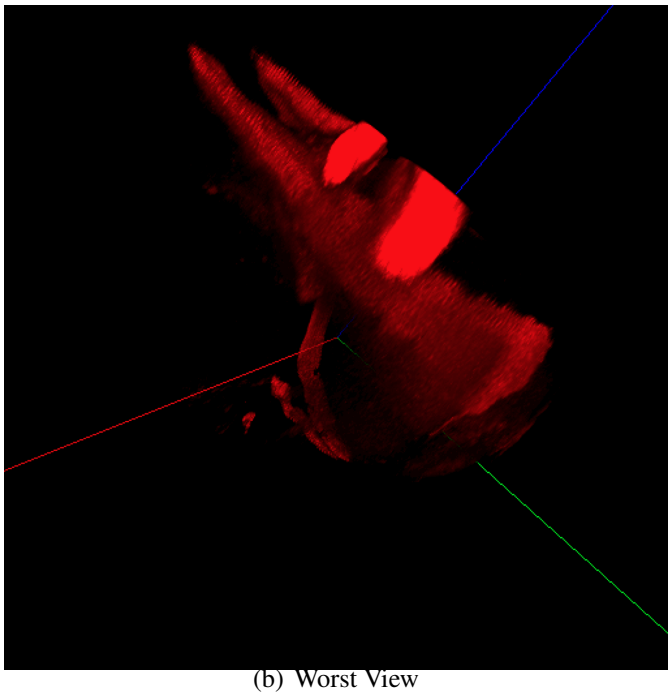
7.4.3 Result

We further apply our algorithm to single channel volume data to compare with other approaches. We select a set of well-known volume data sets in volume rendering communities.

Figure 7.10 shows the results of our algorithm with the Tooth data set. We applied two different transfer functions, to highlight different parts of the data. Figure 7.10(a) and Figure 7.10(b) are the best and worst result for the entire tooth shape. The best direction sets the view from the side allowing viewers to see the entire root of the tooth. The worst direction projects the data upwards. This occludes many interesting parts of the data set, e.g., the shape of the root. The second setting of the transfer function kills the signal in the root, leaving only the crown area of the tooth. Then the interesting part of the data is the shape of the enamel. The best view from our algorithm successfully shows that area in Figure 7.10(c). The worst view in Figure 7.10(d) fails to capture the interesting part of the data. This example illustrates an important point; transfer function design and the viewpoint selection algorithm should be tightly integrated with each other. Transfer functions can change the visual representation of the data set significantly as shown in Figure 7.10, and depending on the setting of the transfer function, the area of interest changes as well. Therefore, the result of the viewpoint selection algorithm can change dramatically with the transfer function. Updating the results in real-time based on the transfer function is critical for understanding the

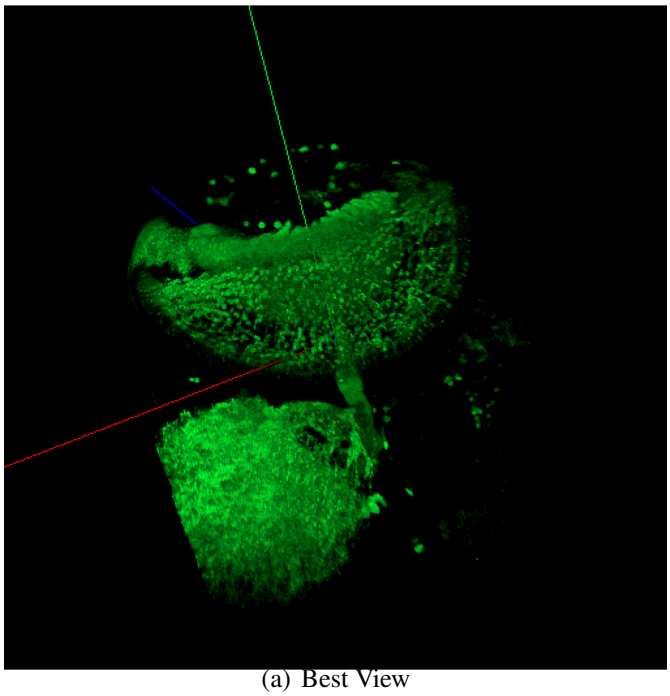


(a) Best View

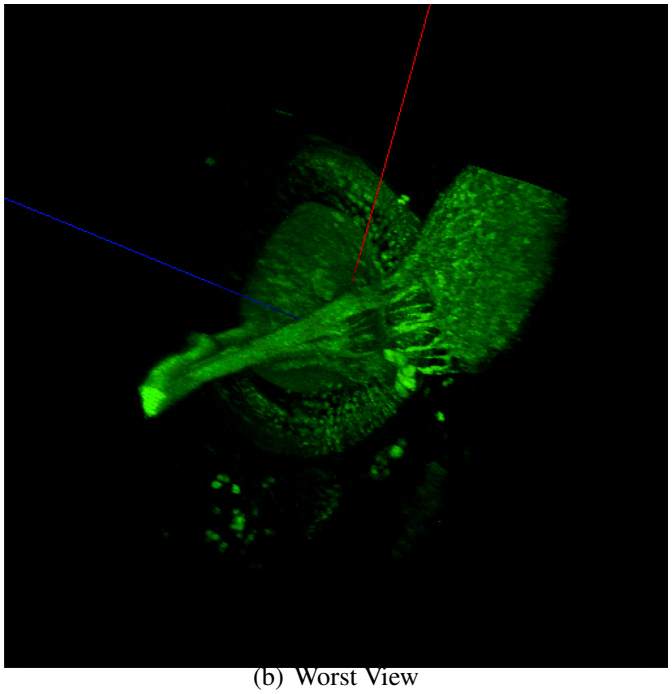


(b) Worst View

Figure 7.7: Results of our viewpoint selection algorithm applied to channel 1 of the zebrafish head data. One can understand the overall structure of the data from the best viewpoint.

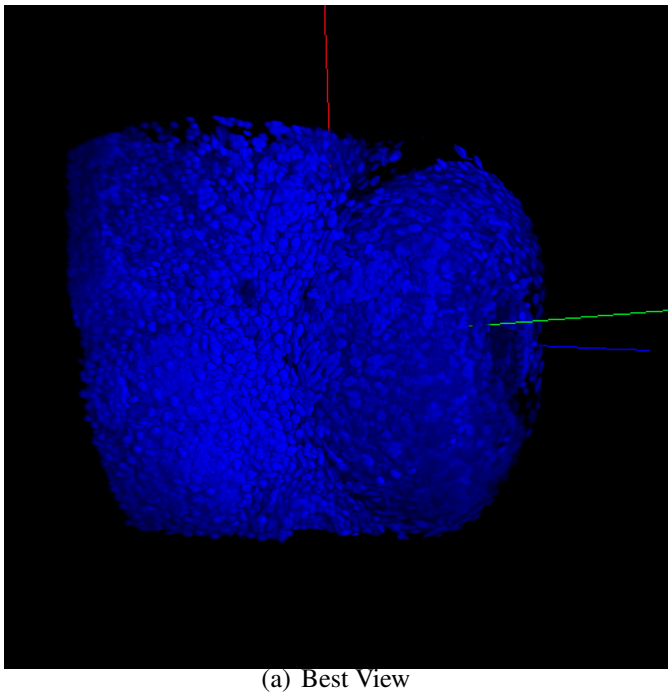


(a) Best View

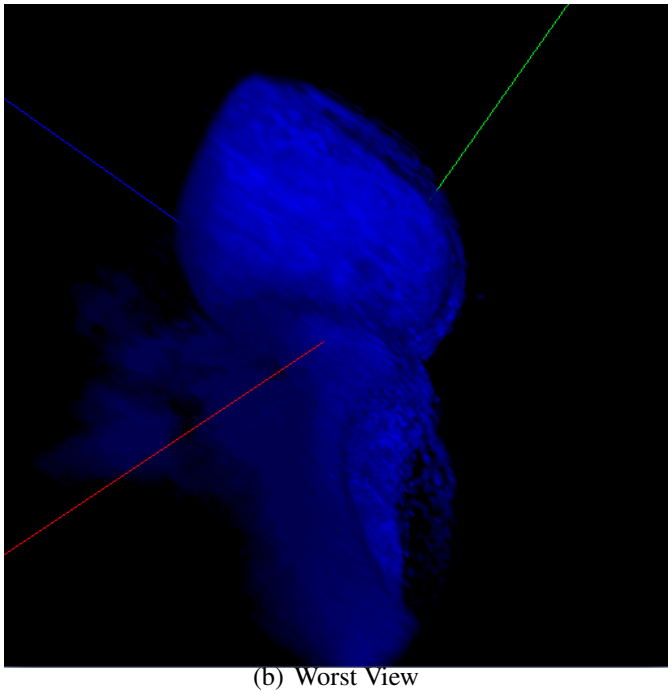


(b) Worst View

Figure 7.8: Results of our viewpoint selection algorithm applied to channel 2 of the zebrafish head data. One can see the shape of tectum and RGCs.

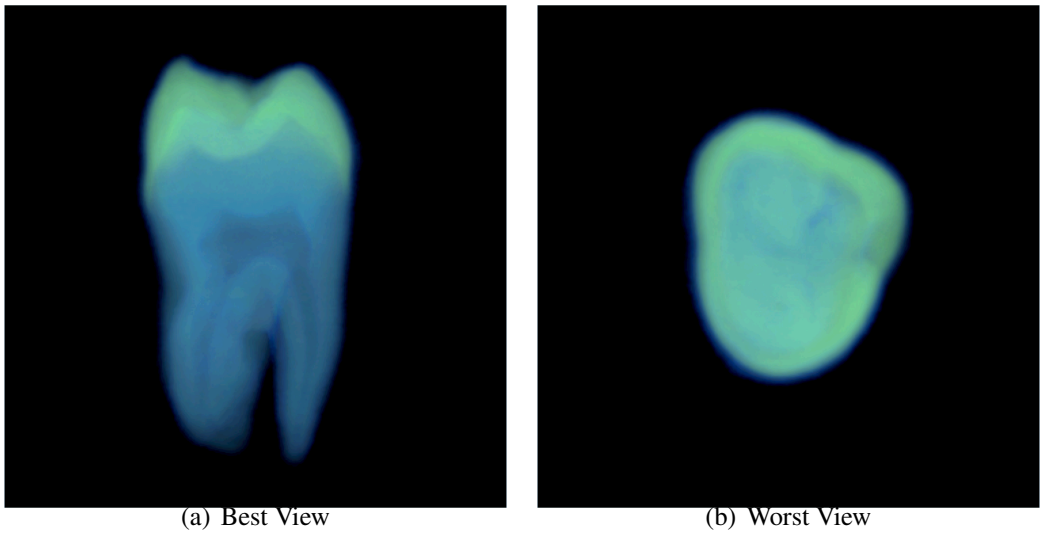


(a) Best View

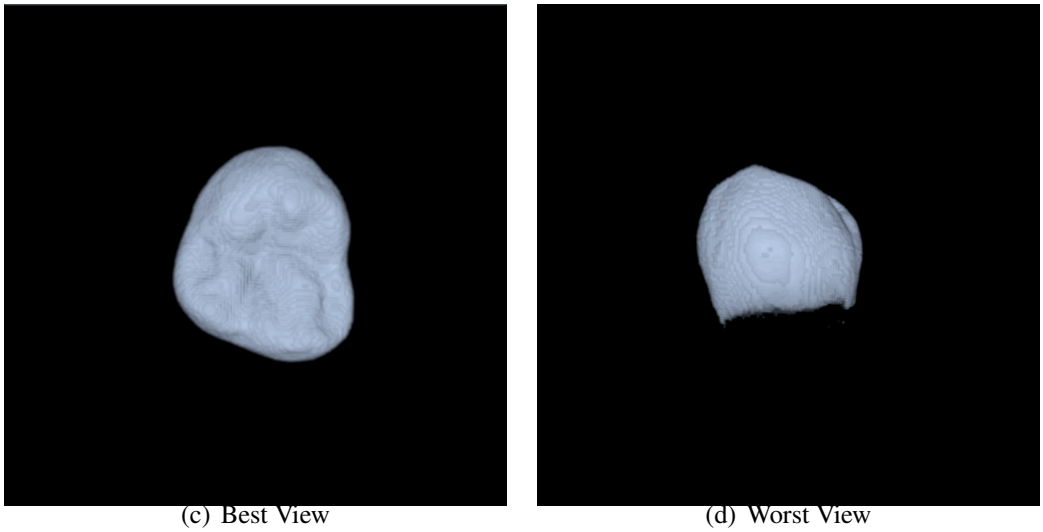


(b) Worst View

Figure 7.9: Results of our viewpoint selection algorithm applied to channel 3 of the zebrafish head data. One can see the cell nuclei on the surface from the best view.

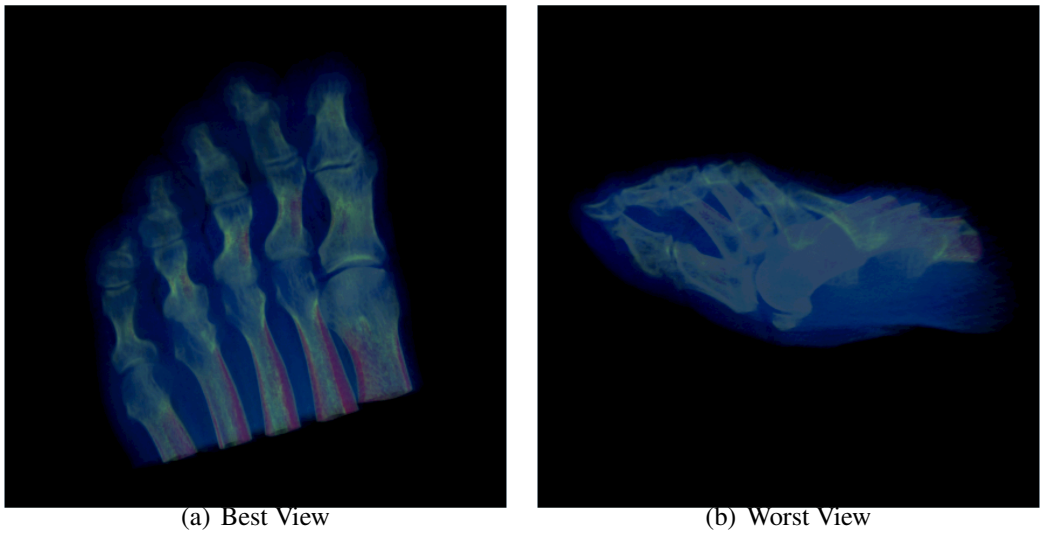


(a) Best View (b) Worst View



(c) Best View (d) Worst View

Figure 7.10: Results for the Tooth data sets with two different transfer function settings. Figure 7.10(a) and Figure 7.10(b) show the best and worst image of the Tooth data set. Figure 7.10(a) shows the entire shape of the tooth by looking at it from the side, while Figure 7.10(b) renders the tooth from the top and it fails to spread out interest points. On the other hand, we modify the volume with a 1D transfer function and render only the crown area without the root of the tooth. In this case, looking from the side does not reveal any particularly interesting information, which is shown in Figure 7.10(d). The most interesting information is the shape of the enamel, and thus viewing from the top shows the interest points more effectively.



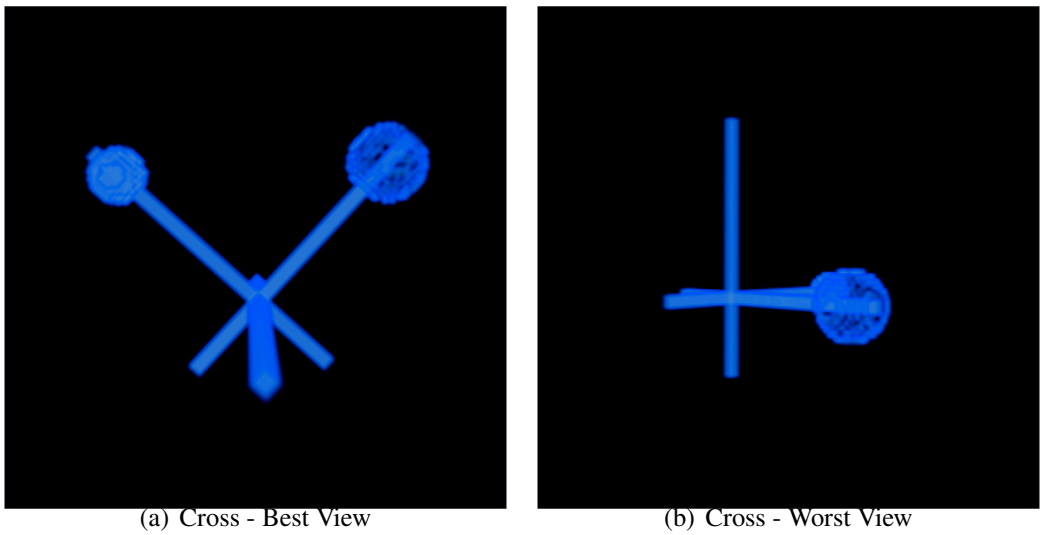
(a) Best View (b) Worst View

Figure 7.11: Results for the Foot CT Scan data from Philips Research, Hamburg, Germany. The best view shows the volume from the top that reveals the overall shape. The worst view renders the volume from another side. Due to the occlusion along the view direction, this view prohibits viewers from understanding the data.

volume data sets.

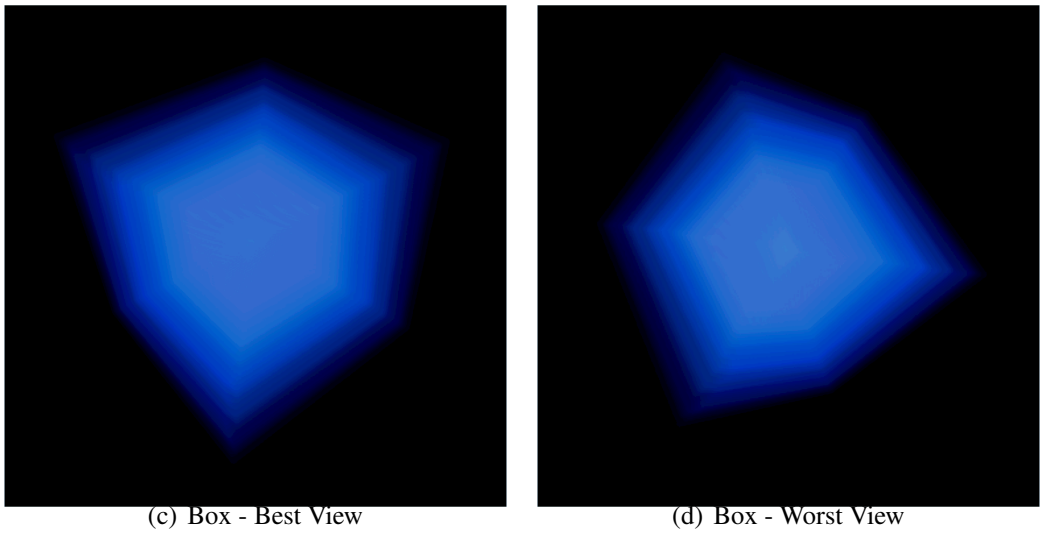
Figure 7.11 shows the Foot CT Scan data from Philips Research, Hamburg, Germany. The data set contains the bones for five toes and soft tissue around them. The features selected by our algorithm are mostly distributed around joints and bones. The variance of these points is maximized when projected on the plane parallel to the top of the foot, as shown in Figure 7.11(a). One cannot see the details if the foot is viewed from the front as in Figure 7.11(b).

Figure 7.12 shows the results for two different data sets. One is the Cross data set and the other is the Box data set, both from the Computer Graphics Group, University of Erlangen, Germany. We set the transfer function so that the Cross data set shows only the three rods and two balls attached at the end of two rods and the Box looks like a cube. The interest points for the Cross data are six ends of the three rods and points in the balls. Therefore, the best view is realized as in Figure 7.12(a). Here we can see the two balls placed far apart although one rod perpendicular to the two rods having balls has to be perpendicular to the view direction. For this data set, one can tell the difference between the best and worst view very easily. However, for the second data set, the Box,



(a) Cross - Best View

(b) Cross - Worst View



(c) Box - Best View

(d) Box - Worst View

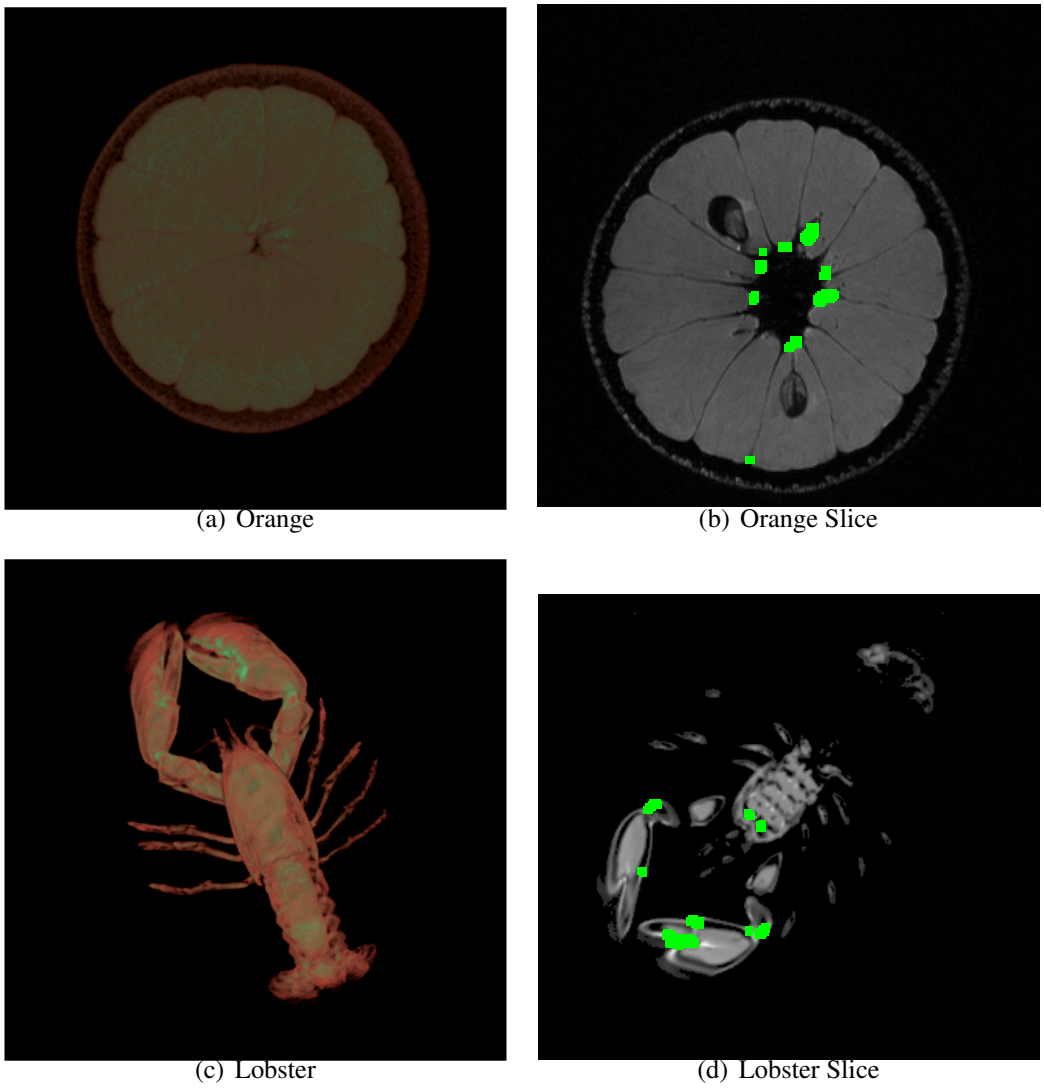
Figure 7.12: The results for the Cross and Box data from the Computer Graphics Group, University of Erlangen, Germany. The transfer function is set to show only the inner part of the Cross data, leaving out outer faces. While the Cross data set shows big difference between best and worst views, it is hard to tell which view works better for the data. The eigenvalue spectrum (shown in Figure 7.14) of these two data sets explains the reason why.

it is not clear which image shows more information than the other due to the symmetric shape.

This can be explained by the spectrum of eigenvalues, and this spectrum is one way of evaluating the quality of the PCA. From the SVD computation, we get three eigenvalues and each eigenvalue is proportional to the variance in that direction. Therefore, if the largest eigenvalue dominates the other two, it means that the variance of the points is mostly captured on the axis defined by the first eigenvector. In contrast, if three eigenvalues are more or less the same, the rendered images from three different directions produce similar results in terms of variances.

This difference is verified by the eigenvalue spectrum in Figure 7.14. For the Cross data set, the first and second eigenvalues occupy 95.6 percent of the sum of the three eigenvalues. This indicates that the plane perpendicular to the third eigenvector, i.e., the view direction, holds 95.6 percent of the variance of the interest points. On the other hand, the Box data set produces three almost identical eigenvalues. This means that the variance of the interest points is equally explained by each axis. No matter where the data is viewed from, it produces equally good images.

Figure 7.13 shows two other data sets. An MRI scan of an orange is from the Information and Computing Sciences Division, Lawrence Berkeley Laboratory, USA. Figure 7.13(b) shows one slice of the data set with the interest points highlighted. Points on the edges of the orange pieces and one seed are selected as interest points. Two bigger seeds are too big to be captured as features by our algorithm. The second data is a CT scan of a lobster. The data is from the VolVis distribution of SUNY Stony Brook, NY, USA. The interest points include two claws and joints as highlighted in Figure 7.13(d). The legs are not selected as interest points because the opacity of the legs is small so the difference in the adjacent voxels is not as big as for other points. The results from the best view directions in Figure 7.13(a) and Figure 7.13(c) display the important features of the objects well.



(a) Orange (b) Orange Slice

(c) Lobster (d) Lobster Slice

Figure 7.13: Additional results. The best view for the Orange data set is from the top. The interest points, highlighted in green, include the center of the orange pieces and a small seed. The Lobster data set is also best displayed from the top, showing all the legs, claws, and the main body. The claws and the joints of the lobster have interest points.

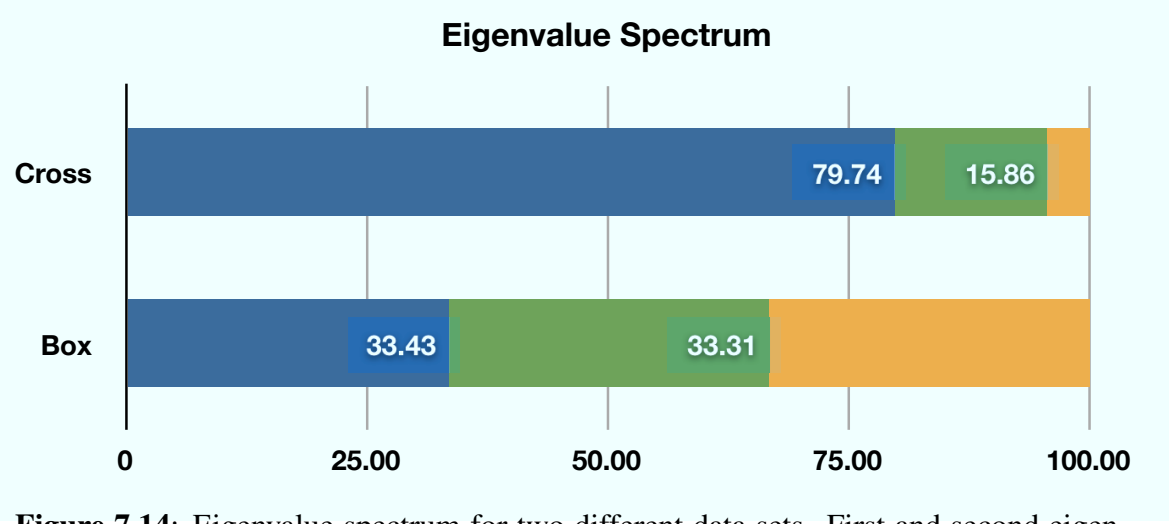


Figure 7.14: Eigenvalue spectrum for two different data sets. First and second eigenvalues of Cross data occupy 95.6% of the three eigenvalues, which means the variance of projected points on the plane created by the first and second eigenvectors explains 95.6% of the entire variance in the original 3D space. Therefore, the projection does not lose much information. On the other hand, the Box data set is symmetrical and there is little difference between best and worst views as shown in Figure 7.12. This is because the first and second eigenvalues add up to only 66.74% and the three eigenvalues equally split the entire 100%. This eigenvalue spectrum tells us how well the best view shows the volume data, and the spectrum gives a quantitative measure for the quality.

7.5 Evaluation

In this section, we compare our algorithm with two previous algorithms as well as with the simplest possible algorithm in terms of performance.

7.5.1 Other Approaches

Information Theory Framework

The approaches based on information theory evaluate the amount of information contained in each view. Bordoloi and Shen [BS05] define the probability distribution function (pdf) on voxels of the data. At each viewpoint, the pdf changes. Ji and Shen [JS06] and Tao et al. [TLB+09] define the pdf on the pixels of the rendered image. Therefore, the evaluation of entropy value requires a rendering process for each view. Because none of the three approaches considers any acceleration method for the search for best viewpoints, the basic mechanism for search is brute-force, i.e., it evaluates as

many points as possible and chooses the best one, i.e., the one that achieves the largest entropy. Then, there is a tradeoff between the computation time and the correctness of the result. If one increases the number of samples, it is possible to find a viewpoint close to the optimal viewpoint. However, this presses the computation time becauses the running time of the algorithm is proportional to the number of rendering, which is very expensive. On the otherhand, if one wants to find a solution in a reasonably short time, the only way is to reduce the number of samples. Then, the average of the angles between samples increase, and it is very unlikely to find the optimal solution.

For the sake of comparison, we use two approaches, one by Ji and Shen [JS06] and the other by Tao et al. [TLB$^+$09]. Ji and Shen use opacity to find which image is better than other images. If there are many pixels that have high opacity in an image, they consider the image to be important. Therefore, they render the volume with opacity data and measure the entropy value of the opacity distribution. The rendered image that has the highest entropy value is the best view. Although they define color and curvature information to measure the quality of a view, it requires multiple rendering process to get entropy values from each measure. This significantly slows the running time, but we were able to get very good results with only opacity, which we discuss in detail in Section 7.5.3.

The algorithm by Tao et al. is based on the same framework but they measure *shape descriptor* and *detail descriptor*. Shape descriptor measures the information that represents the overall structure of a volume data set, while detail descriptor measures the small details of the data. The former is computed from the data created by bilateral filtering [PKT09]. The algorithm filters the original volume, which smooths out noisy information. After the filtering, the resulting volume contains only the overall structure. Then, if a majority of surfaces in the volume is orthogonal to the viewing direction, it means that the volume exposes more surfaces, which they argue provides more information. On the other hand, the difference between the original volume and the filtered volume represents small details that do not affect the overall structure. They also argue that the good view should be able to expose this detail. Therefore, they find a direction where this detail can be maximally exposed. The two pieces of information are combined by a user-defined weight. Because the data we used for our experiments does

not contain local features, we decided to put 0.8 weight on the shape descriptor and 0.2 weight on the detail descriptor. Both descriptors require gradient computation, and the gradient for each fragment is computed in shader.

Area Maximizing Algorithm

We also implement a baseline implementation, the simplest and fastest possible way. We focus more on the performance than the quality of the algorithm. Because volume data sets store data points in a 3D structured grid, the easiest possible way of finding a good viewpoint is to find the direction which the projected area of the grid is maximized. The solution for this algorithm can be computed analytically, and thus the main advantage of this algorithm is the computation time. Only a few arithmetic operations compute the solution. Let w, h, and d denote the width, height, and depth of a volume data set. Then, the direction u that maximizes the projected area of the volume box is defined as follows:

$$u = (\frac{hd}{K}, \frac{dw}{K}, \frac{wh}{K}) \tag{7.10}$$

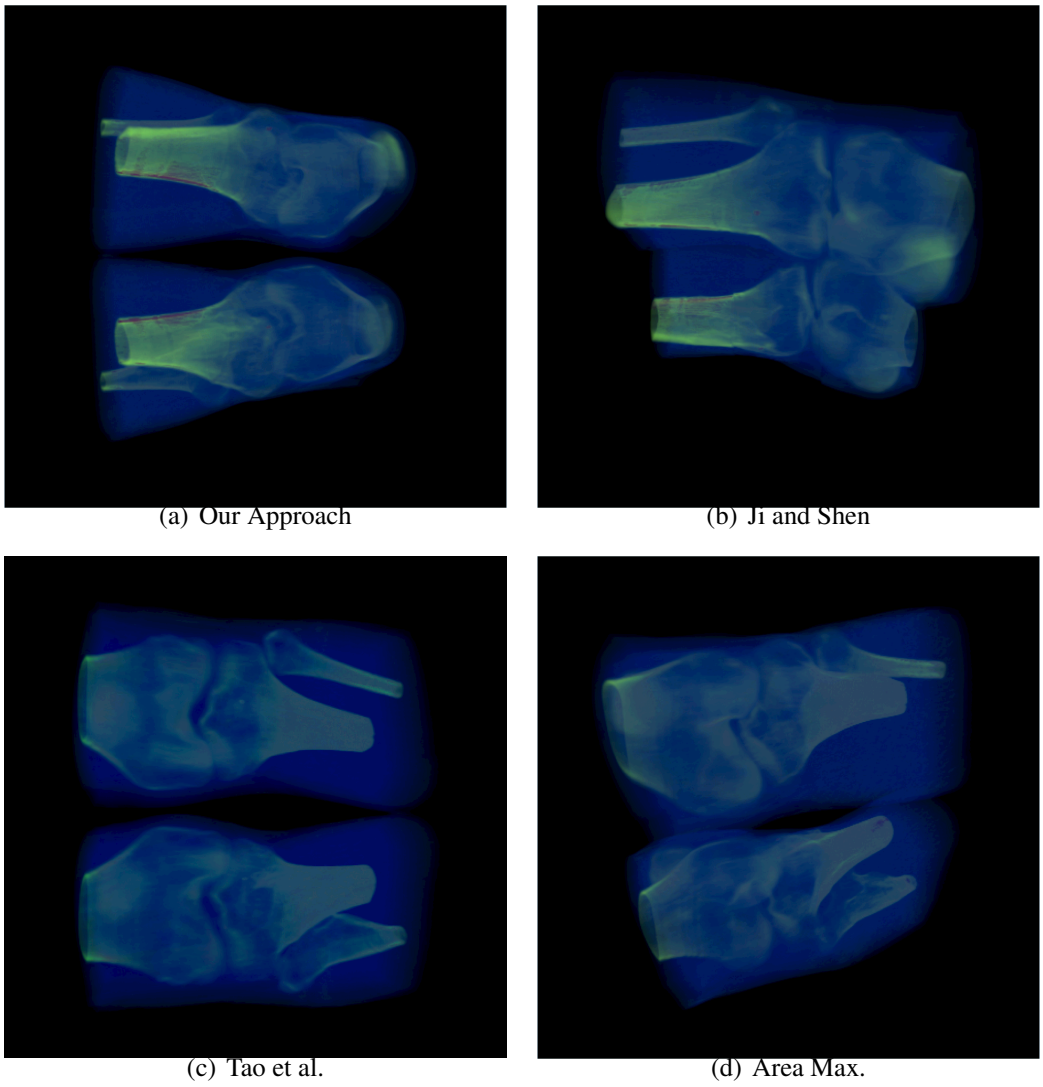
where K is defined as:

$$K = \sqrt{w^2h^2 + h^2d^2 + d^2w^2} \tag{7.11}$$

Despite the obvious advantage, the disadvantage is that the algorithm does not consider what is inside the data set, leading to a poor image quality. There are certainly some cases in which maximizing the bounding box yields a good viewpoint. However, because the algorithm does not know what objects are distributed in the box, it often produces sub-optimal views. In order to alleviate this weakness, instead of computing with the original bounding box, we compute a tighter bounding box to eliminate the empty space in the data.

7.5.2 Performance Comparison

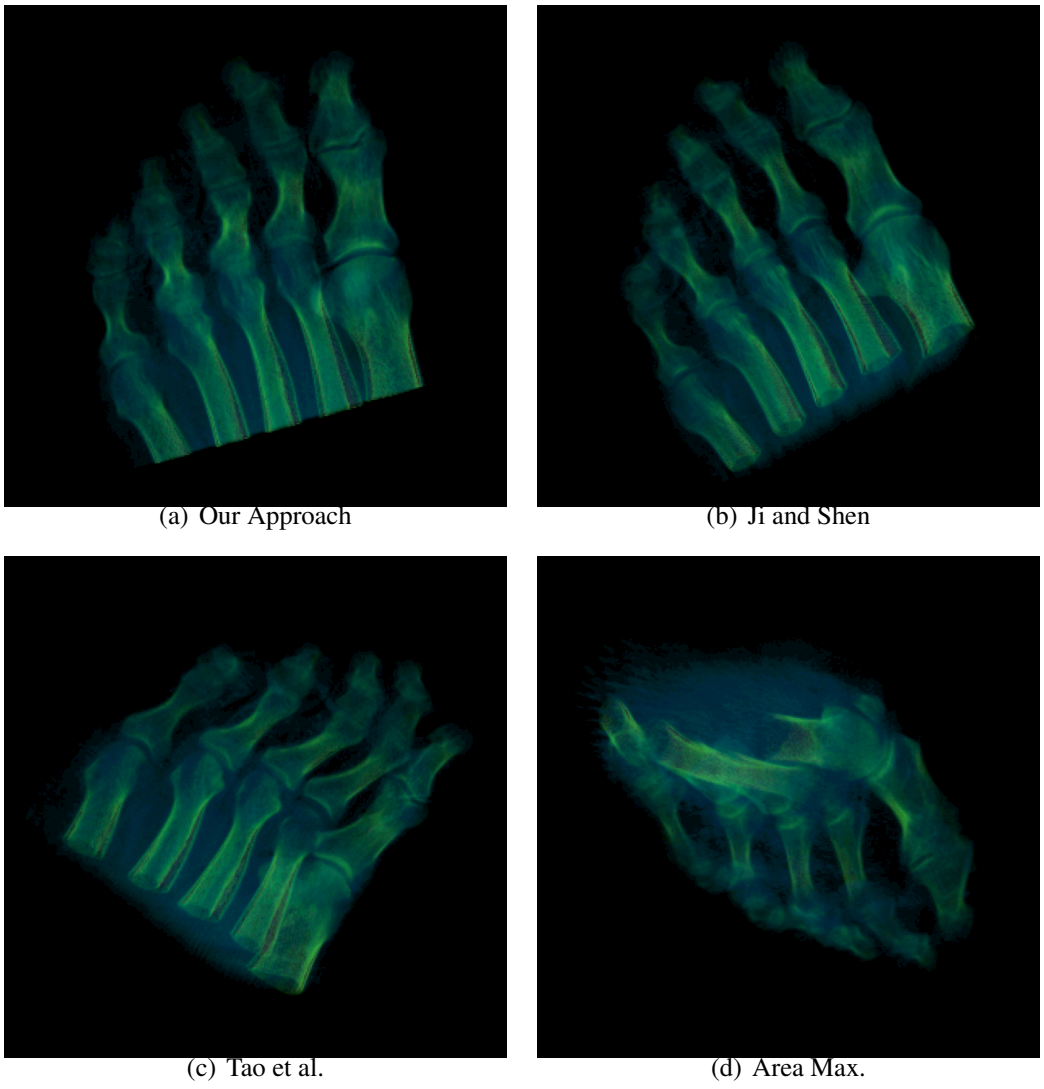
The three approaches (Ji and Shen, Tao et al., and the area maximizing algorithm) were implemented in C++ with OpenGL. The volume rendering algorithm uses



(a) Our Approach (b) Ji and Shen

(c) Tao et al. (d) Area Max.

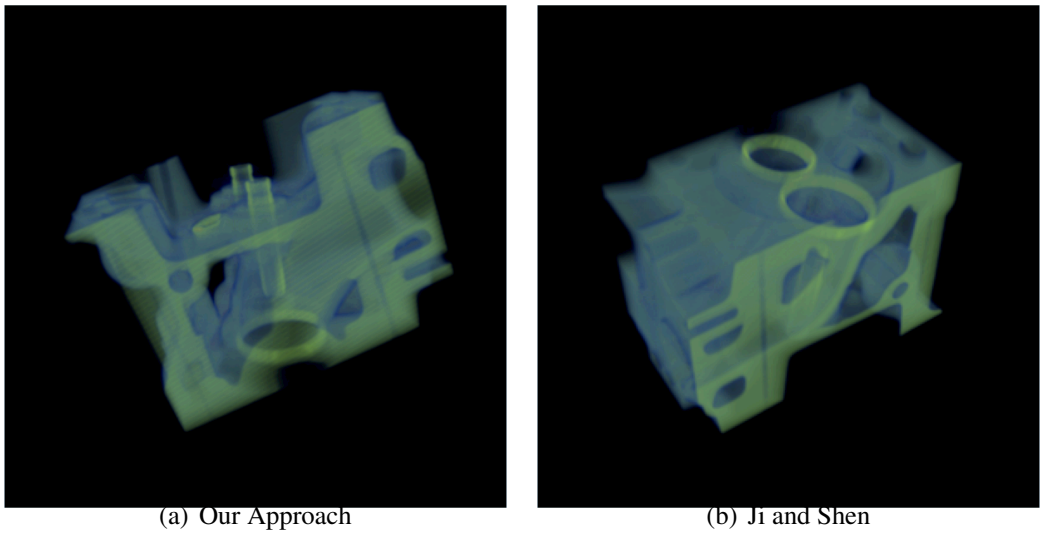
Figure 7.15: The results of our approaches and previous approaches for the knee CT scan data from the Department of Radiology, University of Iowa.



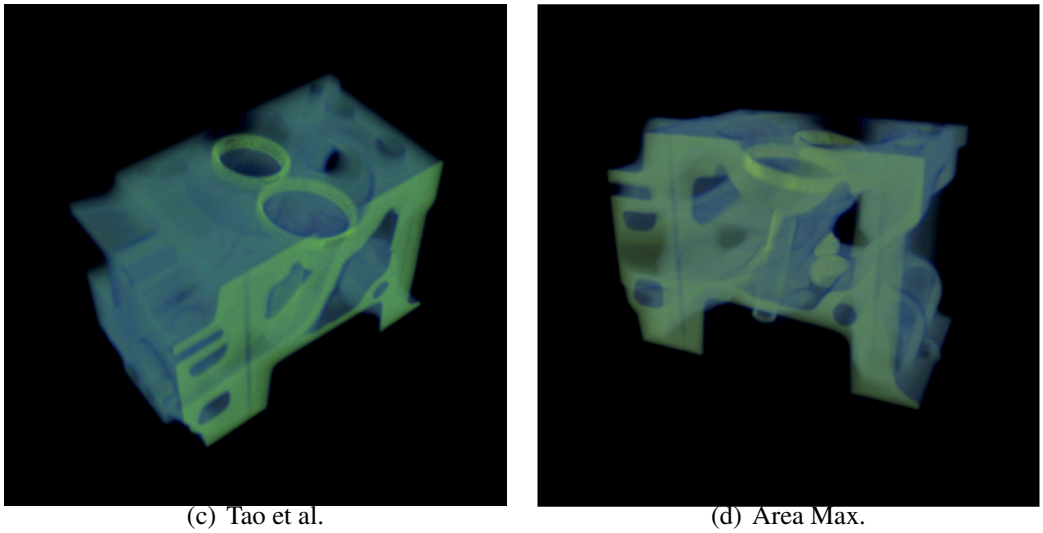
(a) Our Approach (b) Ji and Shen

(c) Tao et al. (d) Area Max.

Figure 7.16: The results of our approaches and previous approaches for the foot CT-scan data.

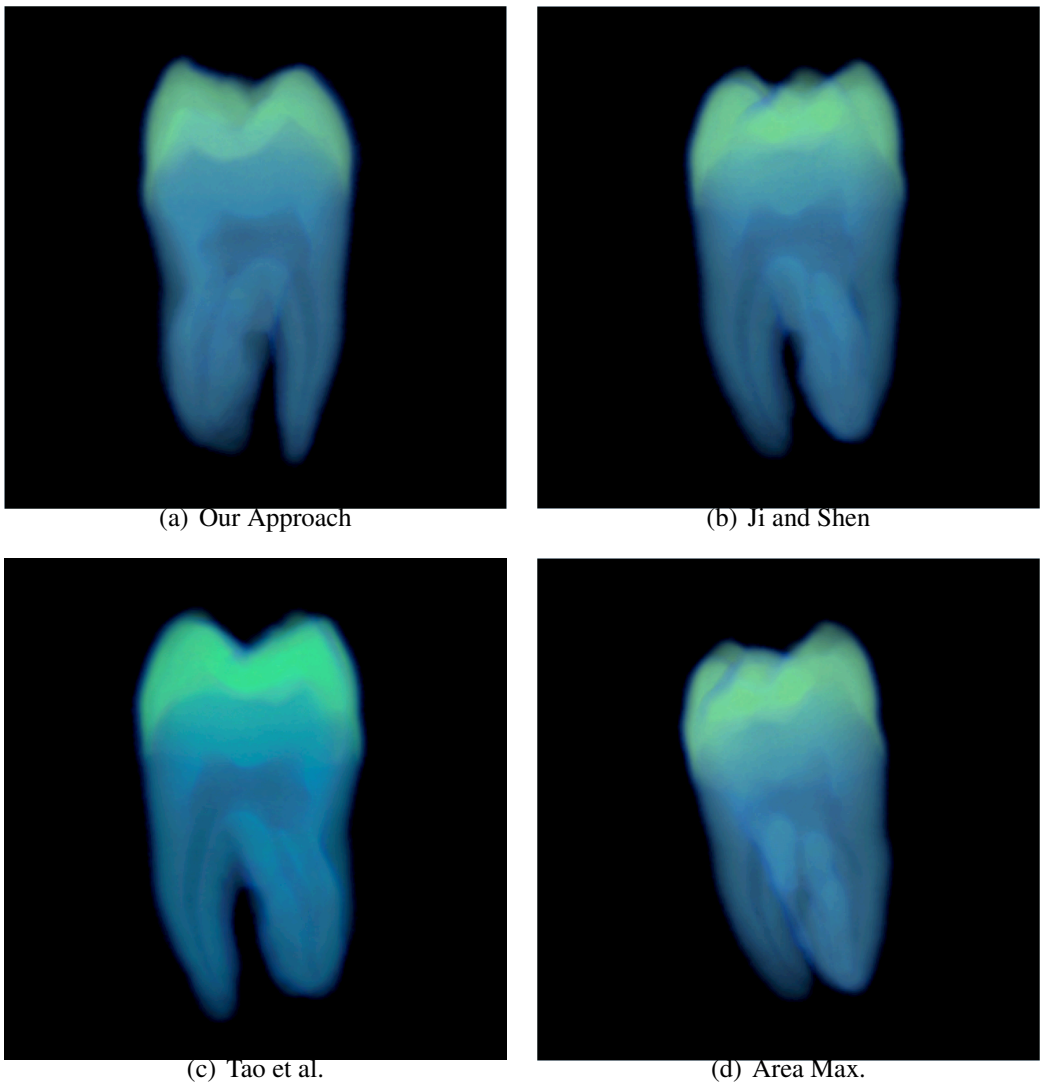


(a) Our Approach (b) Ji and Shen



(c) Tao et al. (d) Area Max.

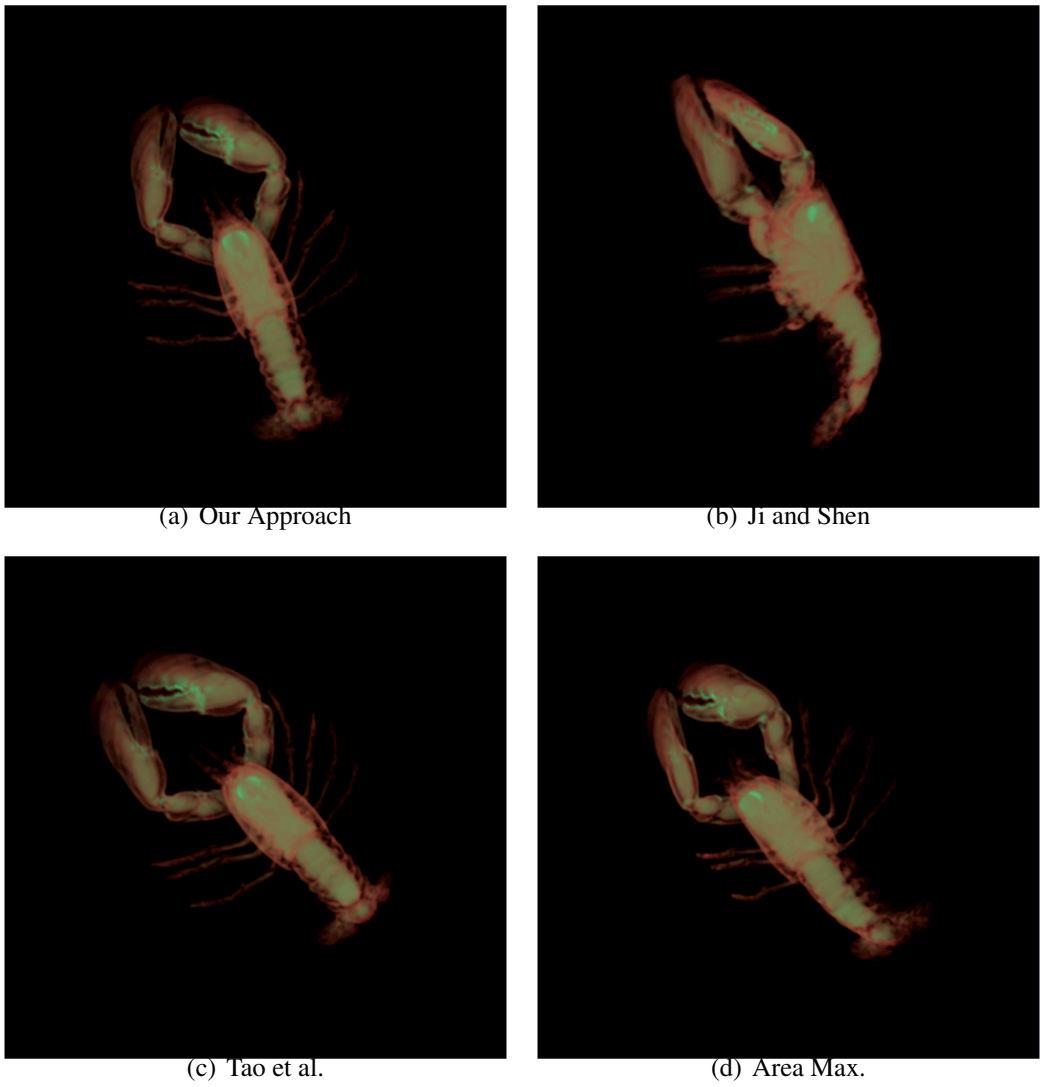
Figure 7.17: The results of our approaches and previous approaches for the engine block data.



(a) Our Approach (b) Ji and Shen

(c) Tao et al. (d) Area Max.

Figure 7.18: The results of our approaches and previous approaches for the tooth data. The tooth data is used in both Ji and Shen [JS06] and Tao et al. [TLB+09]. The data is from the GE Aircraft Engines, Evendale, Ohio, USA



(a) Our Approach (b) Ji and Shen

(c) Tao et al. (d) Area Max.

Figure 7.19: The results of our approaches and previous approaches for the lobster data from the State University of New York Stony Brook, NY, USA.

3D texture and the final image is a result of blending many parallel volume slices. In order to compute the correct view descriptor values for entropy-based approaches, we used GLSL programs. The viewport is set to 512×512 pixels. For the sake of fair comparison, we run all three approaches, as well as ours, on the same machine. The graphics card and CPU in the machine are NVIDIA GeForce GTX 285 and Intel Core i7 2.93 GHz, respectively. During the measurement, we excluded the time for bilateral filtering in the algorithm by Tao et al., which takes more than tens of seconds for most data sets. The computation needs to be included in the comparison, but there are many algorithms proposed [PD09] that are much more efficient than our naive implementation. Furthermore, if one can implement the algorithm in CUDA, it is possible to run in under a second. Therefore, we decided to exclude the timing for the computation, although whenever we re-define the opacity with the transfer function, it is necessary to re-compute the filtering.

Table 7.3 shows the running time of the three algorithms as well as ours. Our approach clearly outperforms Ji and Shen and Tao et al. On average, our approach is 27.16 times faster than the algorithm by Ji and Shen, and 38.66 times faster than the algorithm by Tao et al. The area maximizing algorithm runs 2.20 times faster than our approach. Therefore, in terms of performance, the area maximizing algorithm is the best. However, our approach also runs in a reasonable time and fast enough to run as an interactive component in any volume rendering systems. In the next section, we discuss the quality of the results.

7.5.3 Result Comparison

Figures 7.15, 7.16, 7.17, 7.18, and 7.19 show the results from our algorithm and previous approaches.

In Figure 7.15, we compare the best view for the CT-Knee data. Figure 7.15(a) and 7.15(c) renders the data from the back and the front, respectively. One can easily see that the data shows the symmetric shape of the legs and that it renders the knee of the leg by finding the shapes of bones. Figures 7.15(b) and 7.15(d) also show two legs but the legs slightly overlap each other because the viewpoint is from the side, preventing the viewer from distinguishing the two legs.

Figure 7.16 shows the Foot CT scan data. The viewpoints of Figure 7.16(a) and 7.16(b) are from the top, both of which show the data effectively. However, the viewpoint for Figure 7.16(c) is slightly tilted. One can still understand the shape of the foot in Figure 7.16(c). The worst viewpoint is from the area maximizing algorithm as shown in Figure 7.16(d). In this figure, the bones overlap each other, and it is very difficult to understand the shape of the object.

Figure 7.17 renders the Engine CT scan data. The result of our approach (Figure 7.17(a)) is slightly less descriptive than Figures 7.17(b) and 7.17(c). The area maximizing algorithm generates a decent view, but the entropy based algorithms show more natural images.

For the Tooth data, most algorithms show similar results, except for the area maximizing algorithm, as shown in Figure 7.18. The algorithm shows the tooth from a different angle, failing to separate the two roots of the tooth (See Figure 7.18(d)). In addition, it is hard to see the inside of the tooth due to many occlusions. Other than the area maximizing algorithm, the viewpoint from the side generates a large number of non-zero pixels, (for the algorithm by Ji and Shen) and a large portion of surfaces in the data is exposed (for the algorithm by Tao et al.)

The Lobster data shown in Figure 7.19 favors our algorithm. Our result (Figure 7.19(a)) shows the lobster from the top, revealing all the details from the claws to the tail. Figure 7.19(b) and 7.19(d) show almost identical results where the viewpoint is set slightly tilted to the side. The result from Ji and Shen shows the object from the side. This result is because the algorithm favors an image that has many non-zero pixels. Due to the body and the claw, the projected area of the lobster is maximized.

7.6 Discussion

7.6.1 Applications

Finding a good projection of a 3D data set to a 2D image plane is important in computer graphics, in order to highlight certain features of the data set. As we have shown, our algorithm works well for volume data sets when using transfer functions and alpha blending, partly due to our parallelization for CUDA, which makes our algorithm

Table 7.3: Performance comparison for various volume data sets. We show the running times for various data sets. Entropy-based approaches, Ji and Shen [JS06] and Tao et al. [TLB+09], are significantly slower than our approach and the area maximizing algorithm. Our approach runs in under a second in all data sets, which makes it feasible to be used along with transfer function exploration.

Data Set	Our Approach	Ji and Shen	Tao et al.	Area Maximizing
CT-Knee	0.8128s	6.8281s	18.2919s	1.5686s
Foot	0.5906s	8.4182s	14.0245s	0.8086s
Engine	0.5036s	8.6340s	13.9101s	0.8446s
Lobster	0.2619s	5.6118s	8.2253s	0.1897s
Orange	0.2455s	5.4044s	9.4354s	0.3547s
Tooth	0.1345s	5.0289s	6.5794s	0.0896s
Cross	0.0724s	4.4745s	5.4535s	0.0114s
Box	0.1296s	4.5199s	5.3560s	0.0221s

real-time. Two particularly important features of our algorithm are:

Transfer Function As shown in Section 7.4.2 and Section 7.4.3, transfer function manipulation can dramatically change volume data sets. Different transfer functions may reveal different parts of a data set and thus it is an essential part of volume rendering [PLS+00]. Our viewpoint selection algorithm can be incorporated into the transfer function exploration stage. Whenever a user defines a new transfer function, our viewpoint selection algorithm can update the view direction to best show the features of the data set, so that the user can quickly understand the effectiveness of the new transfer function.

Snapshot for Volume Database Due to the progress in 3D volume acquisition technology, it has become easy to generate many data sets in a short amount of time. These data sets are sometimes published through the web portals [MGW+02b, MTSJ], or put in data banks. The publishers often use snapshots of their volume data sets as proxies to help users find the right data set. It is not practical for the operators of such systems to manually rotate each data set to create good views for the snapshot. Our viewpoint se-

lection algorithm can be used as a snapshot tool to generate proxy images for thousands of volume data sets in a very reasonable amount of time.

7.6.2 Limitation

Despite of the sub-second performance and the simplicity of the algorithm, our approach has three caveats. First, our algorithm assumes that the rendered images would contain the most useful information if the interest points were placed far apart in the rendered images. That works for many data sets. However, the algorithm ignores any occlusion effects. If a data set is rather opaque and the visibility for the inside is limited, it may not be able to see through the volume. In this case, if interest points are selected on the inside, even if the variance of the points is maximized, the result may not produce the best view direction. However, most data sets for direct volume rendering systems are more transparent. If a volume data set is rather opaque, there is not much benefit of rendering a scene with expensive volume rendering process because isosurface rendering can produce better images with more efficient rendering.

Another issue of our approach could be that it uses parallel projection from the 3D data to the 2D image plane to determine the best view direction. In our experience, this works quite well, even if we render the data set with perspective projection, but in some cases the interesting features might not end up in the optimal places when perspective projection is used to render the images.

Finally, PCA gives us three eigenvectors and we pick one as a view direction. However, the eigenvector is an axis, not a direction, on which you can see the interest points effectively. The variance we see from the eigenvector v is the same with $-v$ because all the points are projected to the plane orthogonal to v. For transparent volume data sets, this may not cause a big difference, but one direction may not generate as good an image quality as the opposite direction. One simple solution to this problem is comparing the entropy of the two rendered images to choose the better one. The probability density function for the image can be defined as in many other information theory based frameworks [BS05].

7.7 Summary

We proposed a new approach for finding the best view direction for volume rendering. The algorithm locates interest points in a volume data set and finds a best view in which the interest points are spread out the most. The interest points are found by a 3D extension of the Harris interest point detection algorithm and the view direction is computed by solving a PCA problem. Our algorithm runs significantly faster than other approaches based on information theoretic frameworks. The performance benefit comes from the simplicity of the algorithm and an efficient implementation on the GPU architecture.

Although our algorithm assumes the Harris interest points as important features to look at, it would be interesting to consider different feature selection algorithms, e.g. SIFT [Low99]. Studying other alternatives of the feature selection, in terms of the effectiveness and efficiency, could also be valuable. We would also like to investigate the effect of PCA's linear projection when using perspective projection for rendering. For this goal, we will have to formulate the optimization problem in a different form.

Acknowledgements

This chapter, in part, is based on material that is in preparation for submission: "Real-Time Data-Centric Viewpoint Selection" by Han Suk Kim, Didem Unat, Scott B. Baden, and Jürgen P. Schulze. The dissertation author was the primary investigator and author of this paper.

Chapter 8

Conclusion and Future Work

In this dissertation, we have investigated two ways of improving user experience in volume rendering. Our work has been mostly for multi-channel data in neuro-biology samples, the data of which has the following characteristics:

- **Multiple Values** Each voxel contains multiple values and each value represents the intensity from the fluorescent dye in the specimen. Different proteins are expressed by different types of dye.
- **Distinct Objects** Different channels show distinct objects in the data. Different types of proteins are colocated or complimentary to each other. Scientists are interested in examining how these proteins are distributed in the specimen.

The first characteristic makes the multi-dimensional transfer function design challenging. This is because the transfer function domain easily goes beyond what humans can perceive, i.e., 3D. This problem prevents users from defining a transfer function for multi-channel data although it can be easily defined for single channel data.

Our solution to this problem is to import unsupervised learning algorithms to reduce the dimensionality with minimum loss. Three dimensionality reduction algorithms have been explored: Principal Component Analysis, Isomap, and Locally Linear Embedding. Among the three, the non-linear dimensionality reduction algorithms, Isomap and LLE, have shown that they can capture more variance in the data, which means less information loss when projecting from the original high-dimensional space to the low-dimensional subspace. The original domain constructed from the three intensity values

and derivative values, such as gradient magnitude, are reduced to a three-dimensional distribution. With the low-dimensional distribution, users can define a transfer function as usual. This allows users to understand the multi-channel data quickly and to execute complex classification tasks.

The second characteristic of the multi-channel data requires that users rotate the volume to understand the current appearance of the volume during transfer function exploration. The viewpoints that best describe each channel are dramatically different because, in many cases, each channel shows different objects.

We explored an automatic viewpoint selection algorithm that suggests the best viewpoint for the users. Users can check the overall appearance of the data with minimum rotation because the viewpoint selected by our algorithm displays all the important features of the data with minimum occlusion. This viewpoint suggestion greatly improves the first impression of the data, and thus accelerates the transfer function design. The core idea of our algorithm is to employ a computer vision algorithm to extract important features from the 3D data. From the best viewpoint, the selected features are viewed with minimum occlusion among them. In order to incorporate our algorithm into transfer function design process, we optimized our algorithm with GPU, which improves the performance approximately 10 times more than the implementation on multi-core CPUs.

There are several research directions that will improve the current practice in volume rendering. First, more advanced segmentation algorithms in computer vision and machine learning can improve the transfer function design. Transfer function design is one method of segmenting data in real-time. Using the boundary information or other local information, transfer function helps users to interactively highlight areas of interest. However, direct manipulation of transfer function prevents users from dealing with multi-dimensional information. There are, though, many algorithms proposed in machine learning that retrieve useful information from high-dimensional data. It would be interesting to further investigate how multi-dimensional feature points would impact the transfer function design and what user interface would help users to understand the classification process. As investigated in Maciejewski et al. [MWCE09], advanced clustering algorithms, such as Gaussian mixture model [Bis06], would guide users in

locating areas of interest in the high-dimensional space.

Secondly, it would be interesting to further investigate the way to improve our viewpoint selection algorithm. One possible direction of research is to extend our algorithm so that it can provide a set of best viewpoints. Although other approaches have studied this topic [BS05], it would be interesting to combine automatic segmentation and viewpoint selection. Segmentation finds a set of interesting objects in the data. If each recommended viewpoint best describes a segmented object, users can see the appearance of objects with only a few automatically generated images.

Appendix A

Principal Component Analysis

In this chapter, we review Principal Component Analysis (PCA) in details. We first introduce the problem formulation of PCA and explain how the solution is derived. In addition, we discuss the relationship between PCA and Singular Value Decomposition (SVD).

A.1 Problem Formulation

Let $X \in \mathbf{R}^{N \times D}$ be a matrix where each row contains a sample from D-dimensional space. Without loss of generality, we assume that the samples are centered at the origin. Namely,

$$\sum_{i=0}^{N} X_i = 0$$

where $X_i, (i = 0, \ldots, N-1)$ denotes each row in X. Principal component analysis finds a vector $\hat{u} \in \mathbf{R}^D$ that satisfies the following formula:

$$\begin{aligned} \hat{u} &= \arg\max_{\vec{u}} Var(\vec{u}X) \\ &= \arg\max_{\vec{u}} \frac{1}{N} \sum_{i=1}^{N} (X_i \cdot \vec{u})^2 \end{aligned} \tag{A.1}$$

The solution for Equation A.1 is the first principal component and it is the direction of maximum variance. The second principal component can be obtained as the same

way as in Equation A.1 after subtracting the first component. The number of principal components is D or less and the components are orthogonal to each other. Therefore, PCA is an orthogonal linear transformation where the first coordinate holds the greatest variance of the samples, the second coordinate holds the second greatest variance, and so on.

A.2 Solution

The variance of $\{X_i\}_{i=0}^{N}$ is

$$\begin{aligned} Var(uX) &= E((Xu)^2) \\ &= \frac{1}{N}\sum_{i=0}^{N}(X_i\vec{u})^T(X_i\vec{u}) \end{aligned} \tag{A.2}$$

Therefore, finding the maximum for Equation A.1 is equivalent to finding the maximum value for $\vec{u}^T X^T X\vec{u}$ with respect to $\vec{u}$ where $||u||_2^2 = 1$. In order to find the solution, the Lagrange multiplier method is used.

$$f(\lambda) = \vec{u}^T X^T X\vec{u} + \lambda(1 - \vec{u}^T\vec{u})$$

The maximum value of $f(\lambda)$ is acheived when $\frac{\partial f}{\partial \vec{u}} = 0$.

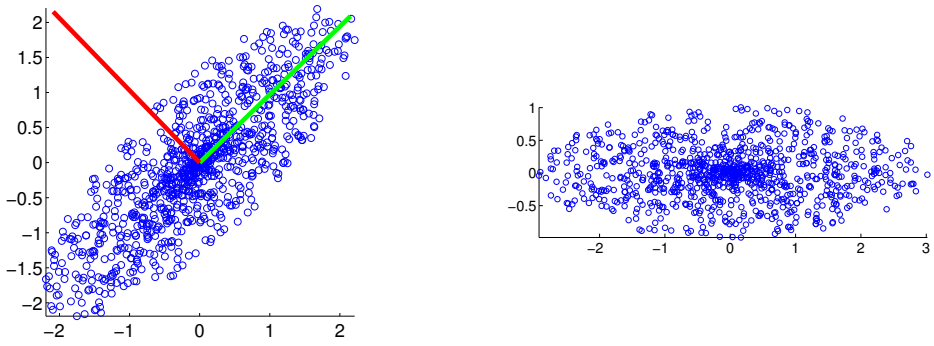
$$\frac{\partial f}{\partial \vec{u}} = 2X^T X\vec{u} - 2\lambda\vec{u} = 0 \tag{A.3}$$

Equation A.3 yields the following relation for the solution:

$$\hat{u} \text{ is a solution to Equation A.1} \Rightarrow X^T X\hat{u} = \lambda\hat{u}$$

Thus, the solution of Equation A.1 is an eigenvector of $X^T X$. We can also show that the eigenvector that corresponds to the largest positive eigenvalue achieves the maximum value for Equation A.1. Moreover, it follows that the eigenvector that corresponds to the second largest positive eigenvalue is the second principal component.

The following figure shows an example of PCA. A set of random points is drawn from an 2D ellipsoid. The main axes for the ellipsoid is rotated with respect to the center



(a) Original Distribution (b) PCA Result

Figure A.1: The original distribution is a random sample from an ellipsoid rotated $+pi/4$ degree. PCA finds the two principal components, which are the first and second eigenvectors of $X^T X$. The two eigenvectors are drawn as green and red lines in Figure A.1(a). The variance of projected points is maximum when the points are projected to the first principal component. The linear transform to these two new axes are shown in Figure A.1(b).

of the coordinate. The two eigenvectors for this sample are $\hat{u}_1 = [0.7175, 0.6965]$ and $\hat{u}_2 = [-0.6965, 0.7175]$. Figure A.1(b) shows the transformed samples with respect to $\hat{u}_1$ and $\hat{u}_2$. The X axis in Figure A.1(b) preserves the maximum variance and the second axes contains the rest of the total variance.

A.3 Dimensionality Reduction

PCA is used as a dimensionality reduction algorithm. The X axis in Figure A.1(b) can explain the original data with a minimum loss, when the data is mapped to one-dimensional space. Because the algorithm linearly projects the data toward the first principal component, if the original samples is not a linear shape, e.g. random samples from a unit circle in 2D or a unit sphere in 3D, the error by the linear projection is significant. However, if the data is spread along a linear line in 2D or plane in 3D, PCA can successfully reduce the dimension to a low-dimensional space with small variance loss.

A.4 Singular Value Decomposition

Finally, PCA has a close relationship with SVD. A matrix $X \in \mathbf{R}^{N \times D}$ can be decomposed into three matrices U, Σ, V such that they satisfy

$$X = U\Sigma V^T \tag{A.4}$$

where $U \in \mathbf{R}^{N \times N}$, $\Sigma \in \mathbf{R}^{N \times D}$, and $V \in \mathbf{R}^{D \times D}$. Each row of matrix U is an eigenvector of XX^T and each row of matrix V is an eigenvector of X^TX. The diagonal values of Σ is eigenvalues of X^TX.

Thus, finding eigenvalues and eigenvectors of the covariance matrix of X in PCA is equivalent to factorize X into U, Σ, and V in SVD:

$$\begin{aligned} X = U\Sigma V^T &\Leftrightarrow X^TX = (V\Sigma U^T)(U\Sigma V^T) \\ &\Leftrightarrow X^TX = V\Sigma^2 V^T \end{aligned} \tag{A.5}$$

Because V has eigenvectors of X^TX, if we can decompose X into $U\Sigma V^T$, we also get the eigendecomposition of X^TX.

The benefit of solving PCA with SVD is that it requires less memory space during computation. Instead of computing $X^TX \in \mathbf{R}^{N \times N}$, which often times creates a huge dense matrix of size N^2, we can work with the original matrix X that has ND elements if $D \ll N$.

Bibliography

[BD90] Kevin W. Bowyer and Charles R. Dyer. Aspect graphs: An introduction and survey of recent results. *International Journal of Imaging Systems and Technology*, 2(4):315–328, 1990.

[BD02] P Bhaniramka and Y Demange. Opengl volumizer: a toolkit for high quality volume rendering of large data sets. *Proceedings of the 2002 IEEE symposium on Volume Visualization and Graphics*, Jan 2002.

[BG97] Ingwer Borg and Patrick J. Groenen. *Modern multidimensional scaling : theory and applications*. Springer series in statistics. Springer, 1997.

[Bis06] Christopher M. Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2006.

[BIT11] BITPLANE: Scientific Software. Imaris, 2011. `http://www.bitplane.com/go/products`.

[BN03] Mikhail Belkin and Partha Niyogi. Laplacian eigenmaps for dimensionality reduction and data representation. *Neural Comput.*, 15(6):1373–1396, 2003.

[BS05] Udeepta D. Bordoloi and Han-Wei Shen. View selection for volume rendering. *Visualization Conference, IEEE*, 0:62, 2005.

[BTB98] Volker Blanz, Michael J. Tarr, and Heinrich H. BÃijlthoff. What object attributes determine canonical views?, 1998.

[CLB05] Laurent Castanie, Bruno Levy, and Fabien Bosquet. Volumeexplorer: Roaming large volumes to couple visualization and data processing for oil and gas exploration. *vis*, 00:32, 2005.

[CLRS01] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction to Algorithms*. MIT Press and McGraw-Hill, 2001.

[CM08] Carlos Correa and Kwan-Liu Ma. Size-based transfer functions: A new volume exploration technique. *IEEE Transactions on Visualization and Computer Graphics*, 14(6):1380–1387, 2008.

[CM09] Carlos Correa and Kwan-Liu Ma. The occlusion spectrum for volume classification and visualization. *IEEE Transactions on Visualization and Computer Graphics*, 15:1465–1472, November 2009.

[CR08] J.J Caban and P Rheingans. Texture-based transfer functions for direct volume rendering. *Visualization and Computer Graphics, IEEE Transactions on*, 14(6):1364–1371, 2008.

[CS99] W Cai and G Sakas. Data intermixing and multi-volume rendering. *Computer Graphics Forum*, 18(3):359–368, Apr 1999.

[CT91] Thomas M. Cover and Joy A. Thomas. *Elements of information theory*. Wiley-Interscience, New York, NY, USA, 1991.

[DCH88] Robert A. Drebin, Loren Carpenter, and Pat Hanrahan. Volume rendering. In *Proceedings of the 15th annual conference on Computer graphics and interactive techniques*, SIGGRAPH '88, pages 65–74, New York, NY, USA, 1988. ACM.

[DG03a] David L. Donoho and Carrie Grimes. Hessian eigenmaps: Locally linear embedding techniques for high-dimensional data. *PNAS*, 100(10):5591–5596, May 2003.

[DG03b] David L. Donoho and Carrie Grimes. Hessian eigenmaps: New locally linear embedding techniques for high-dimensional data, 2003.

[DH92] John Danskin and Pat Hanrahan. Fast algorithms for volume ray tracing. In *Proceedings of the 1992 workshop on Volume visualization*, VVS '92, pages 91–98, New York, NY, USA, 1992. ACM.

[DHS00] Richard O. Duda, Peter E. Hart, and David G. Stork. *Pattern Classification (2nd Edition)*. Wiley-Interscience, November 2000.

[DM98] L. Dagum and R. Menon. OpenMP: an industry standard API for shared-memory programming. *Computational Science and Engineering, IEEE*, 5(1):46–55, Jan-Mar 1998.

[EKE01] Klaus Engel, Martin Kraus, and Thomas Ertl. High-quality pre-integrated volume rendering using hardware-accelerated pixel shading. In *HWWS '01: Proceedings of the ACM SIGGRAPH/EUROGRAPHICS workshop on Graphics hardware*, pages 9–16, New York, NY, USA, 2001. ACM.

[EYSK94] D.S. Ebert, R. Yagel, J. Scott, and Y. Kurzion. Volume rendering methods for computational fluid dynamics visualization. In *Visualization, 1994., Visualization '94, Proceedings., IEEE Conference on*, pages 232 –239, CP26, oct 1994.

[FDM+00] S Fang, Y Dai, F Myers, M Tuceryan, and K Dunn. Three-dimensional microscopy data exploration by interactive volume visualization. *Scanning*, 22(4):218–226, 2000.

[FP02] David A. Forsyth and Jean Ponce. *Computer Vision: A Modern Approach*. Prentice Hall Professional Technical Reference, 2002.

[Fri02] Tobias Friedrich. Nonlinear dimensionality reduction with Locally Linear Embedding and Isomap. MSc dissertation, The University of ShefiňĄeld, Department of Computer Science, September 2002.

[Gar82] Irene Gargantini. Linear octtrees for fast processing of three-dimensional objects. *Computer Graphics and Image Processing*, 20(4):365 – 374, 1982.

[GBKG04] Sören Grimm, Stefan Bruckner, Armin Kanitsar, and Meister Eduard Gröller. Flexible direct multi-volume rendering in interactive scenes. In *Vision, Modeling, and Visualization (VMV)*, pages 386–379, October 2004.

[GWGS02] S Guthe, M Wand, J Gonser, and W Strasser. Interactive rendering of large volume data sets. *Visualization*, Jan 2002.

[Her10] Gabor T. Herman. *Fundamentals of Computerized Tomography: Image Reconstruction from Projections, 2nd edition*. Springer London, second edition, 2010.

[HHKP96] Taosong He, Lichan Hong, Arie Kaufman, and Hanspeter Pfister. Generation of transfer functions with stochastic search techniques. In *VIS '96: Proceedings of the 7th conference on Visualization '96*, pages 227–ff., Los Alamitos, CA, USA, 1996. IEEE Computer Society Press.

[HKG00] J Hladuvka, A Konig, and E Groller. Curvature-based transfer functions for direct volume rendering. *Spring Conference on Computer Graphics*, Jan 2000.

[HM03] Runzhen Huang and Kwan-Liu Ma. Rgvis: region growing based techniques for volume visualization. In *Computer Graphics and Applications, 2003. Proceedings. 11th Pacific Conference on*, pages 355 – 363, oct. 2003.

[HS88] C. Harris and M. Stephens. A combined corner and edge detector. In *Proceedings of the 4th Alvey Vision Conference*, pages 147–151, 1988.

[Jer77] A.J. Jerri. The shannon sampling theorem—its various extensions and applications: A tutorial review. *Proceedings of the IEEE*, 65(11):1565 – 1596, nov. 1977.

[JS06] Guangfeng Ji and Han-Wei Shen. Dynamic view selection for time-varying volumes. *IEEE Transactions on Visualization and Computer Graphics*, 12:1109–1116, 2006.

[KBEGK07] P. Kohlmann, S. Bruckner, M. Eduard Groller, and A. Kanitsar. Livesync: Deformed viewing spheres for knowledge-based navigation. *Visualization and Computer Graphics, IEEE Transactions on*, 13(6):1544 –1551, nov.-dec. 2007.

[KD76] J. J. Koenderink and A. J. Doorn. The singularities of the visual mapping. *Biological Cybernetics*, 24:51–59, 1976. 10.1007/BF00365595.

[KD79] J. J. Koenderink and A. J. Doorn. The internal representation of solid shape with respect to vision. *Biological Cybernetics*, 32:211–216, 1979. 10.1007/BF00337644.

[KD98] Gordon Kindlmann and James W. Durkin. Semi-automatic generation of transfer functions for direct volume rendering. In *Proceedings of the 1998 IEEE symposium on Volume visualization*, VVS '98, pages 79–86, New York, NY, USA, 1998. ACM.

[KGB+09] Bernhard Kainz, Markus Grabner, Alexander Bornik, Stefan Hauswiesner, Judith Muehl, and Dieter Schmalstieg. Ray casting of multiple volumetric datasets with polyhedral boundaries on manycore gpus. In *ACM SIGGRAPH Asia 2009 papers*, SIGGRAPH Asia '09, pages 152:1–152:9, New York, NY, USA, 2009. ACM.

[Kim08] Jusub Kim. *Efficient Rendering of Large 3-D and 4-D Scalar Fields*. Dissertation, University of Maryland, College Park, 2008.

[Kin02] Gorden Kindlmann. Transfer functions in direct volume rendering: Design, interface, interaction. *Siggraph Course Notes*, 2002.

[KKH01] Joe Kniss, Gordon Kindlmann, and Charles Hansen. Interactive volume rendering using multi-dimensional transfer functions and direct manipulation widgets. In *VIS '01: Proceedings of the conference on Visualization '01*, pages 255–262, Washington, DC, USA, 2001. IEEE Computer Society.

[KKH02] Joe Kniss, Gordon Kindlmann, and Charles Hansen. Multidimensional transfer functions for interactive volume rendering. *IEEE Transactions on Visualization and Computer Graphics*, 8(3):270–285, 2002.

[Kru90] Wolfgang Krueger. The application of transport theory to visualization of 3d scalar data fields. In *VIS '90: Proceedings of the 1st conference on Visualization '90*, pages 273–280, Los Alamitos, CA, USA, 1990. IEEE Computer Society Press.

[KSC+10a] Han Suk Kim, Jürgen P. Schulze, Angela C. Cone, Gina E. Sosinsky, and Maryann E. Martone. Dimensionality reduction on multi-dimensional transfer functions for multi-channel volume data sets. *Information Visualization*, 9:167–180, June 2010.

[KSC+10b] Han Suk Kim, Jürgen P. Schulze, Angela C. Cone, Gina E. Sosinsky, and Maryann E. Martone. Multichannel transfer function with dimensionality reduction. In Jinah Park, Ming C. Hao, Pak Chung Wong, and Chaomei Chen, editors, *VDA*, volume 7530 of *SPIE Proceedings*, page 75300. SPIE, 2010.

[KSH01] T. Kohonen, M. R. Schroeder, and T. S. Huang, editors. *Self-Organizing Maps*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2001.

[KW03] J. Kruger and R. Westermann. Acceleration techniques for gpu-based volume rendering. *vis*, 00:38, 2003.

[KWTM03] G Kindlmann, R Whitaker, T Tasdizen, and T Moller. Curvature-based transfer functions for direct volume rendering: methods and applications. *Visualization*, Jan 2003.

[Lev88] Marc Levoy. Display of surfaces from volume data. *IEEE Computer Graphics and Applications*, 8(3):29–37, 1988.

[Lev90] Marc Levoy. Efficient ray tracing of volume data. *ACM Trans. Graph.*, 9(3):245–261, 1990.

[LHJ99] Eric LaMar, Bernd Hamann, and Kenneth I. Joy. Multiresolution techniques for interactive texture-based volume visualization. In *VIS '99: Proceedings of the conference on Visualization '99*, pages 355–361, Los Alamitos, CA, USA, 1999. IEEE Computer Society Press.

[LHWS08] Joshua M. Lewis, Pincelli M. Hull, Kilian Q. Weinberger, and Lawrence K. Saul. Mapping uncharted waters: Exploratory analysis, visualization, and clustering of oceanographic data. *Machine Learning and Applications, Fourth International Conference on*, 0:388–395, 2008.

[Lia07] Chih K. Liang. Bridging the resolution gap: Superimposition of multiple multi-channel volumes. Master's thesis, University of California San Diego, 2007.

[LL03] Ivan Laptev and Tony Lindeberg. Space-time interest points. In *ICCV*, pages 432–439. IEEE Computer Society, 2003.

[LLLa+10] S. Lindholm, P. Ljung, C. LundstroÌĹ andm, A. Persson, and A. Ynnerman. Spatial conditioning of transfer functions using local material distributions. *Visualization and Computer Graphics, IEEE Transactions on*, 16(6):1301 –1310, nov.-dec. 2010.

[Low99] David G. Lowe. Object recognition from local scale-invariant features. In *Proceedings of the International Conference on Computer Vision-Volume 2 - Volume 2*, ICCV '99, pages 1150–, Washington, DC, USA, 1999. IEEE Computer Society.

[LSW+07] Chih K. Liang, Jürgen P. Schulze, Ruth West, Maryann Martone, and Matthias Zwicker. Bridging the resolution gap: superimposition of multiple multi-channel volumes. In *SIGGRAPH '07: ACM SIGGRAPH 2007 posters*, page 121, New York, NY, USA, 2007. ACM.

[LVJ05] Chang Ha Lee, Amitabh Varshney, and David W. Jacobs. Mesh saliency. In *ACM SIGGRAPH 2005 Papers*, SIGGRAPH '05, pages 659–666, New York, NY, USA, 2005. ACM.

[Mac67] J MacQueen. Some methods for classification and analysis of multivariate observations. In *Proc. of the Fifth Berkeley Symposium on Mathematical Statistics and Probability*, pages 281–297, 1967.

[Max95] Nelson Max. Optical models for direct volume rendering. *IEEE Transactions on Visualization and Computer Graphics*, 1(2):99–108, 1995.

[MGW+02a] Maryann E Martone, Amarnath Gupta, Mona Wong, Xufei Qian, Gina Sosinsky, Bertram Ludäscher, and Mark H Ellisman. A cell-centered database for electron tomographic data. *J Struct Biol*, 138(1-2):145–55, Jan 2002.

[MGW+02b] Maryann E Martone, Amarnath Gupta, Mona Wong, Xufei Qian, Gina Sosinsky, Bertram Ludäscher, and Mark H Ellisman. A cell-centered database for electron tomographic data. *J Struct Biol*, 138(1-2):145–55, Jan 2002.

[MHB+00] Michael Meißner, Jian Huang, Dirk Bartz, Klaus Mueller, and Roger Crawfis. A practical evaluation of popular volume rendering algorithms. In *VVS '00: Proceedings of the 2000 IEEE symposium on Volume visualization*, pages 81–90, New York, NY, USA, 2000. ACM.

[MJC02] Benjamin Mora, Jean-Pierre Jessel, and René Caubet. A new object-order ray-casting algorithm. In *VIS '02: Proceedings of the conference on Visualization '02*, Washington, DC, USA, 2002. IEEE Computer Society.

[MM05] K. W. Morton and D. F. Mayers. *Numerical Solution of Partial Differential Equations: An Introduction*. Cambridge University Press, New York, NY, USA, 2005.

[MNTP07] Konrad Mühler, Mathias Neugebauer, Christian Tietjen, and Bernhard Preim. Viewpoint selection for intervention planning. In *EuroVis'07*, pages 267–274, 2007.

[MRH10] Jörg Mensmann, Timo Ropinski, and Klaus Hinrichs. An advanced volume raycasting technique using gpu stream processing. In Paul Richard, José Braz, and Adrian Hilton, editors, *GRAPP*, pages 190–198. INSTICC Press, 2010.

[MTSJ] S. Mikula, I. Trotts, J. M. Stone, and E. G. Jones. Internet-enabled high-resolution brain mapping and virtual microscopy. *Neuroimage*, 35(1):9–15.

[MWCE09] Ross Maciejewski, Insoo Woo, Wei Chen, and David Ebert. Structuring feature space: A non-parametric method for volumetric transfer function generation. *IEEE Transactions on Visualization and Computer Graphics*, 15(6):1473–1480, 2009.

[Nat] National Center for Microscopy Imaging Research. NCMIR.

[NBGS08] John Nickolls, Ian Buck, Michael Garland, and Kevin Skadron. Scalable parallel programming with CUDA. *Queue*, 6:40–53, March 2008.

[Paw89a] J Pawley. Handbook of biological confocal microscopy. 1989.

[Paw89b] James Pawley. *Handbook of biological confocal microscopy*. 1989.

[PBMH05] Timothy J. Purcell, Ian Buck, William R. Mark, and Pat Hanrahan. Ray tracing on programmable graphics hardware. In *SIGGRAPH '05: ACM SIGGRAPH 2005 Courses*, page 268, New York, NY, USA, 2005. ACM.

[PD09] Sylvain Paris and Frédo Durand. A fast approximation of the bilateral filter using a signal processing approach. *Int. J. Comput. Vision*, 81:24–52, January 2009.

[Pea01] K. Pearson. On lines and planes of closest fit to systems of points in space. *Philosophical Magazine*, 2(6):559–572, 1901.

[Per11] PerkinElmer. Volocity, 2011. `http://www.perkinelmer.com`.

[PF08] Francisco de M. Pinto and Carla M. D. S. Freitas. Volume visualization and exploration through flexible transfer function design. *Computers & Graphics*, 32:420–429, August 2008.

[Pho11] EM Photonics. CULA Tools: GPU Accelerated Linear Algebra, 2011. `http://www.culatools.com/`.

[PKT09] Sylvain Paris, Pierre Kornprobst, and Jack Tumblin. *Bilateral Filtering*. Now Publishers Inc., Hanover, MA, USA, 2009.

[PKTD09] Sylvain Paris, Pierre Kornprobst, Jack Tumblin, and Frédo Durand. Bilateral filtering: Theory and applications. *Foundations and Trends in Computer Graphics and Vision*, 4:1–74, 2009.

[PLS+00] Hanspeter Pfister, Bill Lorensen, Will Schroeder, Chandrajit Bajaj, and Gordon Kindlmann. The transfer function bake-off. In *VIS '00: Proceedings of the conference on Visualization '00*, pages 523–526, Los Alamitos, CA, USA, 2000. IEEE Computer Society Press.

[PPB+05] Oleg Polonsky, Giuseppe PatanÃl', Silvia Biasotti, Craig Gotsman, and Michela Spagnuolo. What is an image?: Towards the computation of the "best" view of an object. *The Visual Computer (Proc. Pacific Graphics)*, 21(8-10):840–847, 2005.

[PRC81] S. Palmer, E. Rosch, and P. Chase. Canonical perspective and the perception of objects. *Attention and Performance IX*, pages 131–151, 1981.

[PTCF02] John Plate, Michael Tirtasana, Rhadamés Carmona, and Bernd Fröhlich. Octreemizer: a hierarchical approach for interactive roaming through very large volumes. In *VISSYM '02: Proceedings of the symposium on Data Visualisation 2002*, pages 53–ff, Aire-la-Ville, Switzerland, Switzerland, 2002. Eurographics Association.

[RBS05] Stefan Röttger, Michael Bauer, and Marc Stamminger. Spatialized transfer functions. In *EuroVis*, pages 271–278, 2005.

[RPFC01] Anshuman Razdan, Kamal Patel, Gerald E. Farin, and David G. Capco. Volume visualization of multicolor laser confocal microscope data. *Computers and Graphics*, 25(3):371–382, 2001.

[RS00] Sam T. Roweis and Lawrence K. Saul. Nonlinear Dimensionality Reduction by Locally Linear Embedding. *Science*, 290(5500):2323–2326, 2000.

[RSEB+00] C. Rezk-Salama, K. Engel, M. Bauer, G. Greiner, and T. Ertl. Interactive volume on standard pc graphics hardware using multi-textures and

multi-stage rasterization. In *HWWS '00: Proceedings of the ACM SIGGRAPH/EUROGRAPHICS workshop on Graphics hardware*, pages 109–118, New York, NY, USA, 2000. ACM.

[SB10] Ivan Sipiran and Benjamin Bustos. A robust 3d interest points detector based on harris operator. In *3DOR*, pages 7–14, 2010.

[Sch05] Henning Scharsach. Advanced GPU raycasting. In *Proceedings of the 9th Central European Seminar on Computer Graphics*, May 2005.

[Sch11] Jürgen P. Schulze. DeskVOX: VOlume eXplorer for the Desktop, 2011. `http://www.calit2.net/\~jschulze/projects/vox`.

[SCI] ImageVis3D: A Real-time Volume Rendering Tool for Large Data. Scientific Computing and Imaging Institute (SCI).

[SF90] K. R. Subramanian and Donald S. Fussell. Applying space subdivision techniques to volume rendering. In *Proceedings of the 1st conference on Visualization '90*, VIS '90, pages 150–159, Los Alamitos, CA, USA, 1990. IEEE Computer Society Press.

[SPFG05] Mateu Sbert, Dimitri Plemenos, Miquel Feixas, and Francisco González. Viewpoint quality: Measures and applications. In L. Neumann, M. Sbert, B. Gooch, and W. Purgathofer, editors, *Eurographics Workshop on Computational Aesthetics in Graphics, Visualization and Imaging*, pages 185–192, 2005.

[SR04] Jürgen P. Schulze and Alexander Rice. Real-time volume rendering of four channel data sets. In *VIS '04: Proceedings of the conference on Visualization '04*, page 598.34, Washington, DC, USA, 2004. IEEE Computer Society.

[ST03] Vin De Silva and Joshua B. Tenenbaum. Global versus local methods in nonlinear dimensionality reduction. In *Advances in Neural Information Processing Systems 15*, pages 705–712. MIT Press, 2003.

[TdSL00] J. B. Tenenbaum, V. de Silva, and J. C. Langford. A global geometric framework for nonlinear dimensionality reduction. *Science*, 290(5500):2319–2313, 2000.

[TFTN05] Shigeo Takahashi, Issei Fujishiro, Yuriko Takeshima, and Tomoyuki Nishita. A feature-driven approach to locating optimal viewpoints for volume visualization. *Visualization Conference, IEEE*, 0:63, 2005.

[TLB+09] Yubo Tao, Hai Lin, Hujun Bao, Feng Dong, and Gordon Clapworthy. Structure-aware viewpoint selection for volume visualization. *Visualization Symposium, IEEE Pacific*, 0:193–200, 2009.

[TLM03] Fan-Yin Tzeng, Eric B. Lum, and Kwan-Liu Ma. A novel interface for higher-dimensional classification of volume data. In *VIS '03: Proceedings of the 14th IEEE Visualization 2003 (VIS'03)*, page 66, Washington, DC, USA, 2003. IEEE Computer Society.

[UCB11] Didem Unat, Xing Cai, and Scott B. Baden. Mint: realizing cuda performance in 3d stencil methods with annotated c. In *Proceedings of the international conference on Supercomputing*, ICS '11, pages 214–224, New York, NY, USA, 2011. ACM.

[UKSB11] Didem Unat, Han Suk Kim, Jürgen P. Schulze, and Scott B. Baden. Auto-optimization of a feature selection algorithm. In *Proceedings of the 4th Workshop on Emerging Applications and Many-core Architecture*, June 2011.

[VFSH01] Pere-Pau Vázquez, Miquel Feixas, Mateu Sbert, and Wolfgang Heidrich. Viewpoint selection using viewpoint entropy. In *Proceedings of the Vision Modeling and Visualization Conference 2001*, VMV '01, pages 273–280. Aka GmbH, 2001.

[Vis11] Visage Image. Amira. Website, 2011. `http://www.amira.com`.

[VS03] Pere-Pau Vázquez and Mateu Sbert. Fast adaptive selection of best views. In *Proceedings of the 2003 International Conference on Computational Science and Its Applications: Part III*, ICCSA'03, pages 295–305, Berlin, Heidelberg, 2003. Springer-Verlag.

[WOCH09] Yong Wan, Hideo Otsuna, Chi-Bin Chien, and Charles Hansen. An interactive visualization tool for multi-channel confocal microscopy data in neurobiology research. *IEEE Transactions on Visualization and Computer Graphics*, 15:1489–1496, November 2009.

[WS06] Kilian Q. Weinberger and Lawrence K. Saul. An introduction to nonlinear dimensionality reduction by maximum variance unfolding. In *AAAI*, 2006.

[WSZS07] Kilian Q. Weinberger, Fei Sha, Qihui Zhu, and Lawrence K. Saul. Graph laplacian regularization for large-scale semidefinite programming. In *NIPS*, 2007.

[WWHZ00] M Weiler, R Westermann, C Hansen, and K Zimmermann. Level-of-detail volume rendering via 3d textures. *Proceedings of the 2000 IEEE symposium on Volume Visualization*, Jan 2000.

[YS93] Roni Yagel and Zhouhong Shi. Accelerating volume animation by space-leaping. In *Proceedings of the 4th conference on Visualization '93*, VIS '93, pages 62–69, Washington, DC, USA, 1993. IEEE Computer Society.

[Zeh06] Björn Zehner. Interactive exploration of tensor fields in geosciences using volume rendering. *Comput. Geosci.*, 32:73–84, February 2006.