FLUORENDER, AN INTERACTIVE TOOL FOR CONFOCAL MICROSCOPY DATA VISUALIZATION AND ANALYSIS

by

Yong Wan

A dissertation submitted to the faculty of
The University of Utah
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

in

Computing

School of Computing

The University of Utah

August 2013

The University of Utah Graduate School

STATEMENT OF DISSERTATION APPROVAL

The dissertation of **Yong Wan**

has been approved by the following supervisory committee members:

Charles Hansen	, Chair	**4/29/2013** Date Approved
Christopher Johnson	, Member	**4/29/2013** Date Approved
Gabrielle Kardon	, Member	**4/29/2013** Date Approved
Valerio Pascucci	, Member	**4/29/2013** Date Approved
Ross Whitaker	, Member	**4/29/2013** Date Approved

and by **Alan Davis**, Chair of the Department of **School of Computing**

and by Donna M. White, Interim Dean of The Graduate School.

ABSTRACT

Confocal microscopy has become a popular imaging technique in biology research in recent years. It is often used to study three-dimensional (3D) structures of biological samples. Confocal data are commonly multichannel, with each channel resulting from a different fluorescent staining. This technique also results in finely detailed structures in 3D, such as neuron fibers. Despite the plethora of volume rendering techniques that have been available for many years, there is a demand from biologists for a flexible tool that allows interactive visualization and analysis of multichannel confocal data. Together with biologists, we have designed and developed FluoRender. It incorporates volume rendering techniques such as a two-dimensional (2D) transfer function and multichannel intermixing. Rendering results can be enhanced through tone-mappings and overlays. To facilitate analyses of confocal data, FluoRender provides interactive operations for extracting complex structures. Furthermore, we developed the Synthetic Brainbow technique, which takes advantage of the asynchronous behavior in Graphics Processing Unit (GPU) framebuffer loops and generates random colorizations for different structures in single-channel confocal data. The results from our Synthetic Brainbows, when applied to a sequence of developing cells, can then be used for tracking the movements of these cells. Finally, we present an application of FluoRender in the workflow of constructing anatomical atlases.

For Chi-Bin.

CONTENTS

LIST OF FIGURES

ACRONYMS

Two Dimensional (2D)

Three Dimensional (3D)

Analog-Digital Converter (ADC)

Computed Tomography (CT)

Days Postfertilization (DPF)

Direct Volume Rendering (DVR)

OpenGL Shading Language (GLSL)

Graphics Processing Unit (GPU)

High Dynamic Range Image (HDRI)

High-Resolution Electron Microscopy (HREM)

Laser Scanning Confocal Microscopy (LSCM)

Maximum Intensity Difference Accumulation (MIDA)

Maximum Intensity Projection (MIP)

Magnetic Resonance Imaging (MRI)

Open Graphics Library (OpenGL)

Optical Projection Tomography (OPT)

Personal Computer (PC)

Revolutions Per Minute (RPM)

Serial Advanced Technology Attachment (SATA)

Tagged Image File Format (TIFF)

ACKNOWLEDGMENTS

The FluoRender project started in 2008 after meetings with Charles (Chuck) Hansen, Chi-Bin Chien, and Hideo Otsuna. Although I wrote most of the code of FluoRender, it was the team that made this tool available to many users around the world. Hideo and I used to work around the clock, adjusting the user interface, searching for bugs in the code, and tweaking rendering results. Hideo provided most of the datasets for testing and publications, including, but not limited to: the *Drosophila* visual neurons in Figure 2.2, 4.4, 4.5, and 4.6; the head of zebrafish embryo dataset in Figure 3.3, 3.4, 3.6, 3.9, 3.11, 3.12, 3.14, and 4.7; the tectal neurons of zebrafish embryo in Figure 3.5; the zebrafish eye dataset in Figure 3.8 and 3.13; the mushroom body of *Drosophila* in Figure 3.15; the *Drosophila* brain dataset in Figure 5.5, 5.7, and 5.10; and the *Drosophila* eye dataset in Figure 5.9 and 5.12. Hideo collaborated with Kristen Kwan on zebrafish eye development. Some time-sequence datasets from their experiments are found in Figure 5.8, 5.11, 5.14, and 5.16. It would not be possible for FluoRender to reach its many users without these beautifully imaged confocal scans. Chuck and Chi-Bin's work ensured the funding of the FluoRender project for the past five years. They also provided essential guidance on the development of FluoRender. I shall remember that it was Chi-Bin who gave FluoRender its name. I feel the duty to make FluoRender useful and available to biologists in memory of Chi-Bin.

I later collaborated with Gabrielle Kardon and her group on the anatomical atlas project. Kelsey Lewis generated the confocal scans for the award-winning images as seen in Figure 6.1 and 6.8. Kelsey originally contacted me for the possibility of using FluoRender to present their data, but the results might have turned out to be more fruitful than both of us had expected. I am grateful that I could use my skill and knowledge about digital modeling in scientific research.

Although I have been keeping an interest in digital arts for many years, it was in the University of Utah that I could put it to good use, first as a teaching assistant in the classes of Mark van Langeveld and Robert Kessler, next in the illustrations made for the FluoRender project, and then in the work of the anatomical atlas project. However, the mouse limb atlases would not be possible without all the processing, segmenting, and identifying work done by Kelsey and Mary Colasanto.

I wish to thank other members of my graduate committee. Chris Johnson is the director of the Scientific Computing and Imaging (SCI) Institute, which is the big home that makes all the wonderful work possible. Valerio Pascucci is a renowned scholar whose work I had heard of before he joined SCI. Ross Whitaker taught me about image processing in the first year of my study. Claudio Silva used to be in my committee and he introduced me to scientific visualization.

I feel fortunate to have received help from the research group lead by Chuck. Aaron Knoll advised me on choosing classes when I first came to Utah. Jianrong Shu helped identify Chuck as my advisor. Josh Stratton and Mathias Schott helped read my manuscripts when I started the research. Carson Brownlee provided me help so that I could finally put everything together in this dissertation. It is always a pleasure to meet other members/former members of our small group: Mark Kim, Siddarth Shankar, Liang Zhou, Guoning Chen, and Pascal Grosset.

SCI Institute provides us a comfortable and convenient environment, not just because of the Warnock Engineering Building, but more importantly, because of its wonderful support team members. Chems Touati spent a great amount of time on videos for paper submissions. Deb Zemek, Brenda Peterson, Ed Cask, Magali Coburn, and Tony Portillo are people that I often ask for help. Corinne Garcia, Erik Jorgensen, and Nathan Galli are experts on advertising SCI's research projects, including FluoRender. Certainly, there are not enough words to thank Karen Feinauer and Ann Carlstorm, who help all graduate students of the School of Computing to sort out their administrative headaches.

Finally, all my work was possible because of the funding of the following grants

and foundations: NIH R01-EY12873, Dana Foundation, NSF: CNS-0615194, CNS-0551724, CCF-0541113, IIS-0513212, OCI-0906379, DOE VACET SciDAC, KAUST GPR KUS-C1-016-04, NIH-1R01GM098151-01, NIH R01HD053728 NICHD.

CHAPTER 1

INTRODUCTION

1.1 Motivation

There has been a tremendous explosion in the popularity of confocal microscopy [19] in recent years, due to its ability to scan specimens that have a thickness of hundreds of microns, produce finely-detailed volumetric images, and generate time sequence images of living cells and tissues. In biological research, laser scanning confocal microscopy (LSCM) is an essential tool to study structures of and structural differences between biological samples. Data acquired from confocal microscopy are abundant with finely-detailed structures resulting from fluorescent staining. In order to faithfully reconstruct the 3D structural relationships and enhance the fine details from confocal volumes, specialized visualization tools are demanded by biologists. Furthermore, analysis of confocal data, which focuses on identification and comparison of geometric and topological properties of structures, requires extraction and modeling of those structures under study. There have been a plethora of techniques for volume visualization and segmentation, as well as several academic and commercial packages, but there is always the demand for an interactive tool that integrates carefully selected functionalities and suits general workflows in biological studies with confocal data.

FluoRender is an interactive tool for confocal microscopy data visualization and analysis. It is the result of collaborations between computer scientists and biologists. It is designed and engineered to meet the requirements of biologists. Built upon a slice-based volume rendering kernel, it is capable of reading multiple channels of confocal volumes with a variety of formats,

rendering and mixing channels with different modes, applying 2D image space enhancements, playing back time-sequence confocal data, extracting structures by painting on the volume-rendered results, and visualizing polygon models from those extracted structures along with volumetric data. Despite a tool of many integrated functionalities, FluoRender is striving to provide usability and intuitiveness to its users. As of the time of writing, FluoRender has been available for free download for four years and seen many applications in biological research.

Here, I uncover and present the design, modules, and applications of FluoRender. This chapter continues with the introduction of the three main components of FluoRender: volume visualization, 2D image space enhancements, and interactive extraction of structures. It then introduces Synthetic Brainbows, which is a random colorization technique aiming at identifying structures and tracking cells automatically. The importance and use of anatomical atlases in biological research are then discussed, which lead to our work on the practical workflow of making anatomical atlases from confocal scans.

1.2 Customized Volume Visualization for Confocal Microscopy Data

Most biologists' tools for qualitative analysis of confocal microscopy data are rudimentary, such as looking at image slices or maximal intensity projections. There are several academic and commercial visualization packages available, but these have various significant feature limitations when applied to multichannel confocal data. There is a real demand from biologists for a flexible visualization tool that allows interactive visualization of multichannel confocal data, with rapid fine-tuning of parameters to reveal the three-dimensional relationships of structures of interest.

Confocal microscopy data have their own characteristics, which differ from other biomedical data, such as computed tomography (CT) or magnetic resonance imaging (MRI), which must be taken into consideration as we design such a tool for confocal microscopy visualization.

- Multichannel data. Labeling with different fluorescent proteins and fluores-

cent dyes yields multichannel data, with each channel representing a different cell or tissue type. Usually the data in different channels are spatially interwoven, with data from one channel having the highest interest, such as the channel containing labeled neuron fibers.

• Subtle boundaries. Clearly visualized boundaries of brain regions are often essential for analysis, as when analyzing connectivity of neuron fibers between regions [76, 89]. However, biologically meaningful boundaries may be only subtly presented in the confocal data, and may be present in only one channel of the multichannel data. Thus, boundary segmentation must often be done manually.

• Finely detailed structures. Biomedical techniques such as antibody staining and gene transfer allow delivery of fluorescent dyes to specific cell or tissue types, which can result in very finely detailed structures, such as neuronal fibers or synapses.

• Visual occluders and noise. Structures irrelevant to the analysis may also be labeled through the fluorescent staining process, resulting in visual occluders that obscure the structures to be visualized. Fine detailed structures can also be obscured by noisy data, due to statistical noise or electronic noise from the scanning device [29].

Working together with biologists, we added several enhancements to a slice-based volume renderer and designed a tool for confocal visualization. This tool later became FluoRender. The improvements of this tool on visualizing confocal volumes include the following.

• Interactive settings of volume rendering properties to maximize rendering quality. For better rendering quality and depth perception, we added shading and depth cueing to volume rendering. For detail enhancement and noise suppression, a 2D transfer function can be set through intuitive parameters. All the volume rendering parameters take effect interactively.

• Multimodes for multichannel data visualization. Multichannel datasets can be combined in a single render view with different render modes, with each mode showing a different aspect of the data.

• Embedding polygon data into volume data for region definition. Biological boundaries are usually manually extracted as polygon data with segmentation tools. These polygon data can be rendered together with volume data, which is a clear and efficient way to show the regions of interest. We use depth peeling to achieve correct ordering when semitransparent polygon models are embedded in volumetric data.

1.3 Volume Visualization Enhancements in 2D Image Space

Since its initial release, we continued the development of FluoRender with an emphasis on detail enhancement. The user group of FluoRender has expanded beyond our collaborating biologists. They brought new challenges and problems that we have overlooked in our initial work. One problem that we started looking at were the features presented in 2D image processing packages but commonly missing from volumetric visualization tools. We noticed that most biologists working with microscopy data were actually experts on image processing packages such as Photoshop [3], which were used for a variety of tasks, including combining images, adjusting brightness and contrast, adding annotations, etc. They also have been using tools such as Photoshop with volumetric data visualization results, including those from Maximum Intensity Projection (MIP) [108] and Direct Volume Rendering (DVR) [26, 59]. The familiarity with results from MIP rather than DVR usually makes biologists regard MIP advantageous at rendering sharp details, and this is more common with biologists working with confocal microscopy data, which have an abundance of detail. We have convinced many that DVR can bring out details even better with properly adjusted volume transfer function settings, and will also correctly render the spatial relationship of confocal data. However, users of FluoRender still relied on image processing packages and attempted to enhance details from their retouching work. The retouching work with tools such as Photoshop is usually fraught with frustrations, because the commonly used image formats for data exchange between the visualization tools and image processing packages lack the precision needed for further adjustment,

and these packages are designed for photography rather than confocal data visualization.

We started dealing with the above problems by proposing a series of 2D image space methods for volume visualization enhancement. These techniques were easily integrated with our existing confocal visualization tool. For easier brightness/contrast adjustments and detail enhancement, we used 2D tone-mapping operators, including gamma, luminance, and scale-space equalization. We improved 2D composting for multiple channels by the introduction of groups. To enhance surface details and depth perception, we used 2D compositing to combine a shading and/or a shadow layer with MIP rendering. Similar to our work on volume transfer function, we customized these settings for easy use. These 2D image space methods work in accordance with those volume rendering techniques in 3D data space. They together form the visualization pipeline of FluoRender, which has proved to be useful not only for 3D multichannel confocal scan but also 4D time sequences.

1.4 Interactive Extraction of Biological Structures from Confocal Microscopy Data

In biological research, data analysis focuses on extraction and comparison of geometric and topological properties of structures from confocal microscopy data. FluoRender could generate clear visualizations and facilitate qualitative analysis of confocal microscopy data, but quantitative analysis requires extracting important features. For example, a user may want to extract just one of two adjacent neurons and analyze its structure. In such case, segmentation requires the user's guidance in order to correctly separate the desired structure from the background. There exist many interactive segmentation tools that allow users to select seeds (or draw boundaries) within one slice of volumetric data. Either the selected seeds grow (or the boundaries evolve) in 2D and then user repeats the operation for all slices, or the seeds grow (or the boundaries evolve) three-dimensionally. Interactive segmentation with interactions on 2D slices may be sufficient for structures with relatively simple shapes, such as most internal

organs or a single fiber. However, for most neural structures from confocal microscopy, the high complexity of their shapes and intricacy between structures make even identifying desired structures from 2D slices difficult. 2D-slice-based interactions of most volume segmentation tools become ineffective: it is difficult to choose proper seeds or draw boundaries on slices; even if seeds are chosen and their growth in 3D is automatic, it is difficult for unguided 3D growth to avoid over- or under-segmentation, especially at detailed and complex structures such as axon terminals; there is no interactive method to quickly identify and correct the segmented results at problematic regions. With well-designed visualization tools, biologists are able to observe the complex neural structures and inspect them from different view directions. Segmentation interactions that are designed based on volume visualization tools and let users select from what they see are apparently the most intuitive. In practice, confocal laser scanning can generate datasets with high throughput, and biologists often conduct experiments and scan multiple mutant samples in batches. Thus, a segmentation algorithm for structure extraction from confocal data also needs to make good use of the parallel computing power of contemporary personal computer (PC) hardware and generate a stable segmented result with real-time speed.

To help analysis on confocal microscopy data, we added interactive segmentation functions into FluoRender. It uses morphological diffusion for region-growing, which can generate stable results for confocal data in real-time; its interaction scheme explores the visualization capabilities of our existing confocal visualization pipeline, and lets users paint directly on volume rendering results and select desired structures. A close integration of visualization and segmentation techniques within one tool allows biologist users to extract structures of interests from their visualization workflow. On the other hand, segmentation further improves visualization results by removing occluding structures or emphasizing important structures.

1.5 Synthetic Brainbows

Brainbow [63] is a genetic engineering technique that randomly colorizes cells. Biological samples processed with this technique and imaged with confocal microscopy have distinctive colors for individual cells. It is useful for disambiguating visual clutter in confocal data, identifying complex cellular structures, and for cell tracking. However, the application of the Brainbow technique on a certain species of animals requires complex transgenic manipulations. In practice, most confocal microscopy scans use different antibody staining with typically at most three distinct cellular structures. These structures are often packed and obscure each other in rendered images, making analysis difficult. The problem is commonly addressed through segmentation. Accurate segmentation of confocal microscopy data, which are typically full of fine details, depends greatly on users' prior knowledge of the data. Such knowledge does not only come from experience, but also is more and more importantly from visualizations of the data. Visualizing confocal microscopy data requires techniques that on the one hand are interactive and preserve the fine details on the other. Inspired by the Brainbow technique, we explored techniques that randomly colorize single-channel confocal microscopy data. The outcome of our random colorization technique assumes similar appearance of Brainbows. Adjacent complex structures can then be clearly visualized by color variations. Our Synthetic Brainbow technique leverages a process known as GPU framebuffer feedback loops, which is a random process in the massive parallel computing environment of GPUs. In addition, we incorporated ID shuffling and Monte-Carlo sampling into the technique. The random colorization in our synthesized Brainbow images respects structural information and preserves fine details. The results were presented to domain experts with positive feedback. A user survey demonstrated that our Synthetic Brainbow technique improved visualization of volume data with complex structures for biologists.

1.6 Making Anatomical Atlases

Anatomical atlases of humans and other species are important scientifically for understanding normal anatomy, the development and function of these structures, and for determining the etiology of congenital abnormalities. Unfortunately, for biologists, it is difficult to generate such atlases, especially ones with the informative content and aesthetic quality that characterize human atlases. Building such atlases requires the knowledge of the species under study and experience with an art form that can faithfully record and present this knowledge, both of which require extensive training in fields considerably different from one another. With the latest innovations in data acquisition and computing techniques, atlas building has changed dramatically. It is now possible to create atlases from 3D images of biological specimens, allowing for high-quality, faithful representations. Labeling of structures using fluorescently-tagged antibodies, confocal three-dimensional scanning of these labeled structures, volume rendering, segmentation, and surface reconstruction techniques all promise solutions to the problem of building atlases. However, biological researchers still ask the question, "Is there a set of tools we can use or a practical workflow we can follow so that we can easily build models from our biological data?" The question is heard mostly by computer scientists and answered by a vast number of algorithms, tools, and program codes. Most of these computer scientists are able to tackle one aspect of the problem or provide solutions to some special cases. Nevertheless, the general question of how to build anatomical atlases remains unanswered. For a satisfactory answer, biologists need a practical workflow that can be easily adapted for different applications. Second, reliable tools must be readily available that can fit into the workflow. Lastly, examples using the workflow and tools to build anatomical atlases would demonstrate the utility of these resources for biological research.

We designed a generalized workflow to generate anatomical atlases from confocal microscopy scans. The workflow is adapted from a CG artist's workflow of building 3D models for animated films and video games. Having been developed, tested, and employed for industrial use for decades, the artist's

workflow with certain adaptations is the most suitable for making high-quality anatomical atlases, especially under strict budgetary and time limits. FluoRender is used in this workflow along with artists' tools, such as Maya [5] and Mudbox [6]. We demonstrate how FluoRender is used in practice for biological research with this detailed case study.

1.7 Thesis Statement

Confocal microscopy data visualization and analysis are real and important applications in biology research, which require the integration of both existing and novel techniques in volume rendering, image processing, user interaction, and digital arts.

1.8 Thesis Contributions

The main contributions of this work are the following.

• Detailed descriptions of a close collaboration between computer scientists and biologists as well as a tool developed through the collaboration. Many of the techniques presented in this work may seem unconventional to experts in visualization, such as the render modes for multiple channel rendering. This is because our biologist collaborators gave us prompt comments and suggestions during the whole process of development. On the other hand, we also learned new techniques from biologists. For example, the Synthetic Brainbow technique is inspired by the Brainbow technique, which is a genetic engineering technique that randomly colorizes cells. With the increase of interdisciplinary collaborations, we believe there will be more examples like ours in the future.

• Practical methods that solve real-world problems in scientific visualizations. Since FluoRender has always been a publicly-released tool for biologist users, effective and intuitive solutions are always demanded. This is the reason that we integrated a full set of 2D image space method into our system for visualization enhancement. Even though we are dealing with datasets with ever-increasing dimensions, simple methods at lower dimensions shall never be overlooked, as they may be solutions to complex problems.

• Improvements to user interactions with volumetric data. For transfer function manipulations, widgets previously used in 2D histograms are replaced with parametrized settings that are specially designed for confocal data. Segmentation of 3D structures can be carried out directly in the rendered results. These techniques simplify operations so that more data can be processed with shorter time.

• Integration of artists' tools into the workflow of making and visualizing anatomical atlases from confocal scans. Although many artists' tools did not originate from scientific research, their data processing power and especially usability may exceed tools used in science. Incorporating artists' tools into scientific visualization workflow also makes it possible for artists to participate in scientific research. The combined creativity of science and art may lead us to new findings.

1.9 Outline

In Chapter 2, background knowledge of confocal microscopy and related work in volume visualization and analysis are discussed. Chapter 3 details the visualization pipeline of FluoRender, which is an integration of volume rendering techniques and 2D image space enhancements. Chapter 4 presents the interactive volume segmentation methods in FluoRender, which uses a paint brush analogy and allows user-guided extraction of structures. Chapter 5 is one step forward towards the automation of the methods discussed in Chapter 4. We developed a technique that simulates the random colorization in the well-known Brainbow technique. The random colorization is used to disambiguate complex structures in confocal data, especially the cells. Chapter 6 is a real-world application of FluoRender in biological research, where FluoRender plays a key role in a workflow of making anatomical atlases from confocal microscopy data. Finally, conclusions and future work are given in Chapter 7.

CHAPTER 2

BACKGROUND

2.1 Laser Scanning Confocal Microscopy

Laser scanning confocal microscopy has become an invaluable tool for a wide range of investigations in the biological and medical sciences for imaging thin optical sections in living and fixed specimens. The basic concept of confocal microscopy was developed by Marvin Minsky in the mid-1950s [19]. The concept was continually perfected in the 1970s and 1980s, which led to the first commercial product in 1987 [19]. Coupled to the rapidly advancing computer processing speeds, enhanced displays, and large-volume storage technology emerging in the late 1990s, laser scanning confocal microscopy finally gained tremendous popularity in the last decade in many applications.

A modern confocal microscope is a completely integrated electronic system consisting of an optical microscope, several laser systems, electronic detectors, and a computer. The principal components of its optical system are the objective lens system, a dichromatic mirror, and two pinhole apertures (Figure 2.1). The two pinhole apertures are positioned at the conjugate points to the focus point of the objective lens. The pinhole aperture at the laser source diffracts the laser beam, which is then focused on a point within the specimen. The specimen, usually fluorescently stained, emits fluorescent light of a different wavelength from the excitation laser. The fluorescence emission from the focus point is refocused by the objective lens at the second pinhole aperture. A significant amount of fluorescence emission that occurs at out-of-focus points is not confocal with the second pinhole aperture and is cut off from being detected by the electronic detector. Therefore, only a small point, usually at submicrometer level, of the specimen is imaged at

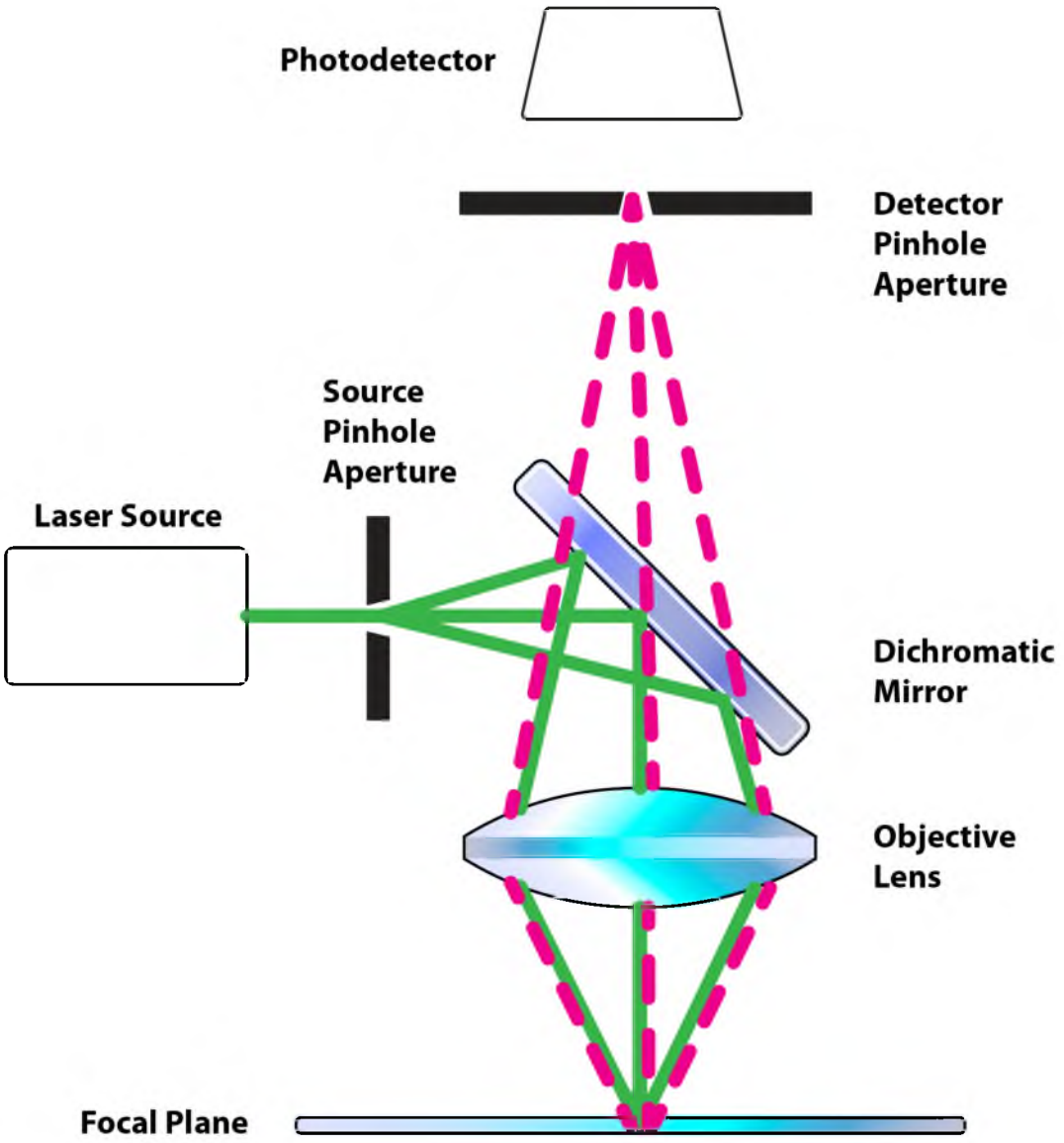

Figure 2.1. Principle components of the optical system of a laser scanning confocal microscope. The excitation light (green) and emitted light (red) have different wavelengths. The emitted light rays are cut off by the detector pinhole aperture so that only a single point in the specimen is scanned a time.

a time. This imaging process repeats within the focus plane (lateral XY axes) or deep into the specimen (Z axis) and a three-dimensional image is scanned.

A confocal microscope is commonly equipped with several laser sources and lasers of different wavelengths can be used to excite the fluorescently-stained specimen. Different antibodies that bind only to particular structures (for example, neurons, muscles, and tendons) are used to deliver fluorescent tags. A confocal microscope excites the fluorescent tags with lasers, and different detectors collect the emitted light from the tags. So for example, one detector will collect the image of tendons in green, another will collect muscles in red, and another will collect neurons in blue.

Compared to conventional widefield fluorescent microscopy, the primary advantage of laser scanning confocal microscopy is the ability to capture three-dimensional structures with fine details, without physically sectioning the specimen. This noninvasive optical sectioning technique enables the examination of both fixed and living specimens, resulting in volumetric data and time-sequence data.

2.2 Interactive Visualization of Volumetric Data

Volume rendering is a set of techniques for visualizing volumetric data typically generated by CT and MRI scanners, or from computer simulations. Since previous research has largely solved the rendering problems of volumetric data, recent research of volume visualization has shifted its focus on the following: firstly, methods that facilitate accurate interpretation and extraction of meaningful information from complex, abstract, and multidimensional data in different applications; secondly, interactive techniques scalable for extremely large datasets, including multichannel, multimodal, and time-sequence data; thirdly, interaction and integration among volume visualization, other forms of visualization and analysis, for example, polygon mesh generation and rendering, unstructured data visualization, statistical and topological analysis, etc.

Confocal microscopy data visualization is an application topic in volume

visualization, which involves research of more or less all three points above. As in the introduction chapter, multiple channels of confocal data naturally pose a question of how different channels are combined and rendered. Cai and Sakas [15] proposed three levels of data intermixing and rendering pipelines in direct multivolume rendering, which include image level intensity intermixing, accumulation level opacity intermixing, and illumination model level parameter intermixing. In the context of radiotherapy treatment planning, they applied their methods to three volumes, including CT volume, dose volume, and segmentation volume. Then they compared the features of different data intermixing methods. Rossler et al. [85] described a flexible framework for GPU-based multivolume rendering, which provided a correct overlaying of an arbitrary number of volumes and allows the visual outputs for each volume to be controlled independently. They also presented a visualization tool specific for the rendering of functional brain images, which was built on top of their frame work. Their tool included different GPU-based volume rendering techniques, on the one hand for the interactive visual exploration of the data, and on the other hand for the generation of high-quality visual representation. Grimm [36] presented a full-blown high-quality raycasting system, which can efficiently process and visualize multiple large medical volume datasets. The core acceleration technique of his system was a refined caching scheme for gradient estimation in conjunction with a hybrid skipping and removal of transparent regions to reduce the amount of data to be processed. In addition, the system distinguished regions where multiple volumes intersect, and efficiently rendered regions containing only one volumetric object, and those need costly multivolume processing.

The fine details, noise, and visual occluders in confocal microscopy data require visualization techniques that can visualize important structures without explicit segmentation. Volume classification has been an active area of research. Kindlmann et al. [50] proposed the histogram volume, which captures the relationship between volumetric quantities in a position-independent, computationally efficient fashion. Then they presented semi-automatic methods of generating

transfer functions for direct volume rendering. Kniss et al. [53] presented multidimensional transfer functions for interactive volume rendering. Their work demonstrated an important class of three-dimensional transfer functions for scalar data. It described the application of multidimensional transfer functions to multivariate data. They also presented a set of direct manipulation widgets that made specifying such transfer functions intuitive and convenient. Correa et al. [21, 23] proposed visibility-driven and size-based transfer function designing techniques for volume exploration. They incorporated visibility histogram and size information into a multidimensional transfer function design. Their semi-automated method for transfer function generation progressively explored the transfer function space towards the goal of maximizing visibility of important structures. To ease the difficulties for end-users to use multidimensional transfer functions, methods have been proposed to accelerate the transfer function design process. Rezk-Salama et al. [82] introduced an additional level of abstraction for parametric models of transfer functions. They proposed a framework that allowed visualization experts to design high-level transfer function models that can intuitively be used by nonexpert users. The resulted user interface provided semantic information for specialized visualization problems. Tzeng et al. [100] proposed an approach to the volume classification problem that couples machine learning and a painting metaphor to allow more sophisticated classification in an intuitive manner. Their intelligent system approach enables users to perform classification in a much higher dimensional space without explicitly specifying the mapping for every dimension used.

Embedding polygon data into volumetric data is one technique to visualize subtle boundaries and the content within each boundary. Everitt [28] described an algorithm for interactively rendering order-independent transparent polygon objects with graphics hardware. The algorithm is also known as depth peeling. The depth peeling algorithm is widely used for correctly blending transparent polygon meshes. Kreeger and Kaufman [55] presented an algorithm that embeds opaque and/or translucent polygons within volumetric data, by rendering thin slabs of the

translucent polygons between volume slices using slice-order volume rendering. They demonstrated their algorithm with examples of medical applications and flight simulators. Nagy and Klein [70] presented the concept of volumetric depth-peeling, and they separated the volume data into interior and exterior based on a fixed iso-value. Weiskopf et al. [111] proposed clipping methods that are capable of using complex geometries for volume clipping, which enable selecting and exploring subregions of one volumetric dataset.

There are some commercially available software packages for confocal data. Amira [106] can render volume datasets from confocal microscopes, and visualize them together with polygon data, which are generated by its segmentation tool automatically or manually. Imaris [10] incorporates multiple volume rendering algorithms for visualizing microscopy data interactively, and it can also generate polygon data for rendering or volume editing. Volocity [79] can load multichannel confocal data, and it provides both interactive and noninteractive volume renderers for visualizing them. For specific problems and data, users often feel problems with these tools: many do not provide adequate parameter settings for fine-tuning volume rendering results; some are not interactive when adjusting parameters; and it is always laborious to analyze repetitive experiments.

2.3 Visualization Enhancement in 2D Image Space

2D image space methods are processing methods applied after the volumetric data are projected and rendered into the 2D image space, such as 2D filtering, tone mapping, and compositing. In the application domain of volume visualization, most 2D image space methods can be carried out more efficiently than their 3D counterparts. Most importantly, 2D image space methods can be used to enhance volume visualization quality when applied together with volume rendering methods. Most research of volume visualization enhancement focuses on algorithms in 3D data space. For example, Ebert and Rheingans [27] introduced a volume illustration approach, which is a combination of the familiarity of a physics-based illumination model with the ability to enhance important features

using nonphotorealistic rendering techniques. They included properties such as volume sample location and value, gradient value, view direction, and light information into the volume illustration procedure. The features that could be enhanced include boundaries, silhouettes, depth and orientation cues, distance color bleeding, halos, and tone shading. Kuhn et al. [56] designed an image-recoloring technique and applied it to volume rendering for highlighting important visual details. To improve visualization experiences for individuals with color vision deficiency, Machado et al. [66] proposed a physiologically-based model for recoloring visualization results, including volume-rendered scientific data. For illustrative visualization, Wang et al. [110] presented a framework to aid users to select colors for volume rendering, in which case color mixing effects usually limit the choice of colors. While there are several commercial and academic visualization packages that biologists have been using for confocal microscopy data, such as Amira [106], Imaris [10], and Volocity [79], 2D image space methods for detail enhancement are generally absent from these tools. In fact, choosing and designing proper 2D image space methods and parameters, integration of 2D and 3D methods, as well as their applications are interesting research topics for volume visualization in general. In [13], Bruckner et al. presented a framework for compositing of 3D renderings, and use the framework for interactively creating illustrative renderings of medical data. Tikhonova et al. [98] [99] proposed visualization by proxy, which is a framework for visualizing volume data that enables interactive exploration using proxy images. For fast prototyping and method/parameter searching, computer scientists often use comprehensive visualization and image processing libraries, such as VTK [52] and ITK [51]. Experimental applications with customized visualization pipelines are generated. However, this is often regarded as impractical by biologist users, since they usually demand a reliable tool with seamlessly integrated functions that are only relevant to their specific application scenario.

2.4 Volume Segmentation Techniques

In biomedical research, useful segmentation methods for volumetric data are generally categorized into two kinds: full manual and semi-automatic. The concept of fully automatic segmentation does exist; however, either the implementations are limited to ideal and simple structures, or they require complex parameter adjustment, or a vast amount of manually segmented results are used for training. They fail in the presence of noisy data, such as confocal scans. Thus, robust fully automatic segmentation methods do not exist in practice, especially in cases where complex and intricate structures are extracted according to users' research needs.

In biology research, fully manual segmentation is still the most-used method. Though actual tools vary, they all allow selecting structures from each slice of volumetric data. For example, Amira [106] is often used for extracting structures from confocal data. For complex structures, such as neurons in confocal microscopy data, it requires great familiarity with the data and the capability of inferring 3D shapes from slices. For the confocal dataset shown in Figure 2.2, it took one neurobiologist one week to manually select one neuron, since it was difficult to separate the details of the two neurons in proximity. However, such intense work would not guarantee a satisfactory result: some fine fibers of low scalar intensities might be missing. Even when the missing parts could be visualized with a volume rendering tool, it was still difficult to go back and track the problems within the slices. To improve the efficiency of manual segmentations, biologists have tried different methods. For example, VolumeViewer from Sowell et al. [96] allows users to draw contours on oblique slicing planes, which helps surface construction for simple shapes but is still not effective for complex structures. Using the volume intersection technique from Martin and Aggarwal [68] or Space Carving from Kutulakos and Seitz [57], Tay et al. [97] drew masks from two orthographic MIP renderings and projected them into 3D to carve a neuron out from their confocal data. However, the extracted neuron in their research had a very simple shape.

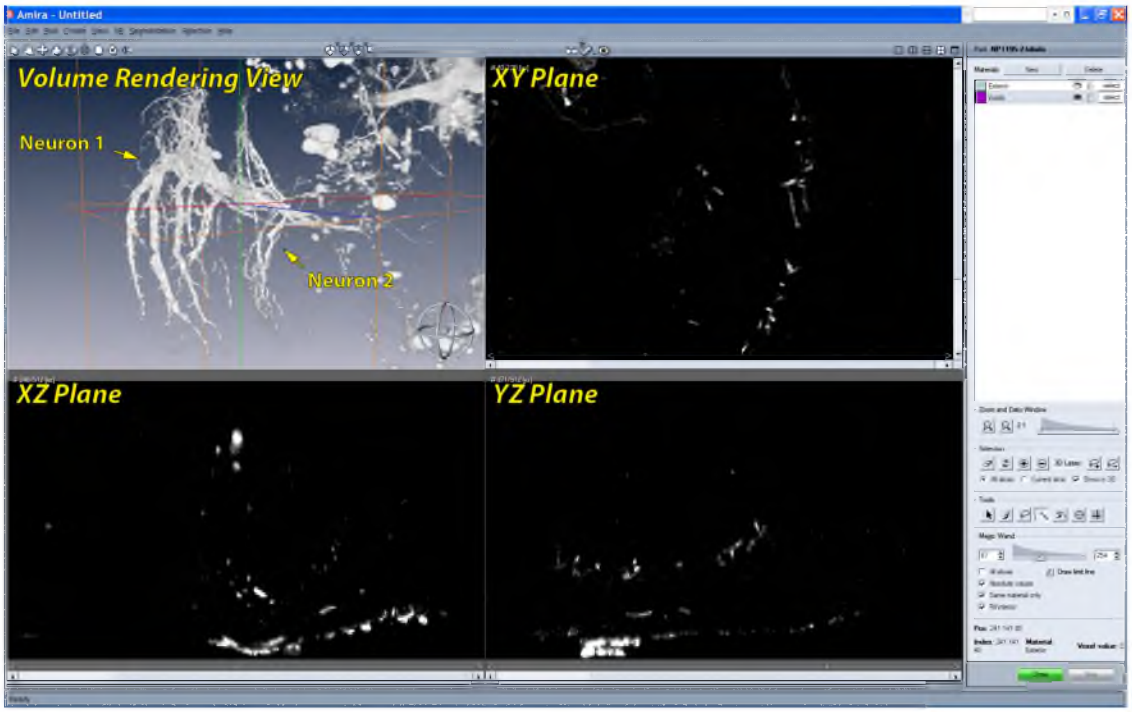

Figure 2.2. The user interface for segmentation in Amira. In the volume rendering view, we can observe that two neurons are in proximity and have complex details. However, it is difficult to tell them apart or infer their shapes from any of the slice views. Unfortunately, users have to select structures from the slice views rather than the volume rendering view, where they can actually see the data more clearly. Many interactive volume segmentation tools in neurobiology use similar interactions, which are difficult to use for complex shapes.

For extracting complex 3D structures, semi-automatic methods, which combine specific segmentation algorithms with user guidance, seem to be more advantageous than manual segmentation. However, choosing an appropriate combination of algorithm and user interaction for a specific segmentation problem, such as neural structure extraction from confocal data, remains an active research topic. Though the variety of segmentation algorithms is myriad, many of them for extracting irregular shapes consist of two major calculations, i.e., noise removal and boundary detection. Most filters designed for 2D image segmentation can be easily applied for volumetric data. Filters commonly seen include all varieties of low-pass filters, bilateral filters, and rank filters (including median filter, as well as dilation and erosion from mathematical morphology) [33]. Boundaries

within the processed results are very commonly extracted by calculations on their scalar values, gradient magnitudes, and sometimes curvatures. Prominent segmentation algorithms that see many practical applications in biology research include watershed [104], level set [75], and anisotropic diffusion [80]. The latest technological advances in graphics hardware allow interactive application of many previously proposed algorithms to volumetric data. Sherbondy et al. [93] implemented anisotropic diffusion on programmable graphics hardware and applied the framework to medical volume data. Viola et al. [105] implemented nonlinear filtering on graphics hardware and applied it to segmenting medical volumes. Lefohn et al. [58] implemented the level-set algorithm on graphics hardware and demonstrated an interactive volume visualization/segmentation system. Jeong et al. [44] applied the level-set method to EM datasets, and they demonstrated an interactive volume visualization/segmentation system. Hossain and Möller [41] presented an anisotropic diffusion model for 3D scalar data, and used the directional second derivative to define boundaries. Saad et al. [88] developed an interactive analysis and visualization tool for probabilistic segmentation results in medical imaging. They demonstrated a novel uncertainty-based segmentation editing technique, and incorporated shape and appearance knowledge learned from expert-segmented images [87] to identify suspicious regions and correct the misclassification results. Kniss and Wang [54] presented a segmentation method for image and volume data, which is based on manifold distance metrics. They explored a range of feature spaces and allowed interactive, user-guided segmentation.

Most segmentation research has focused on improving accuracy and robustness, but little has been done from the perspective of user interactions, especially in real-world applications. Sketch-based interaction methods, which let users directly paint on volume rendering results and select desired structures, have demonstrated the potential towards more intuitive semi-automatic volume segmentation schemes. Yuan et al. [113] presented a method for cutting out 3D volumetric structures based on simple strokes that are drawn directly on

volume rendered images. They used a graph-cuts algorithm and could achieve near-interactive speed for CT and MRI data. Chen et al. [18] enabled sketch-based seed planting for interactive region growing in their volume manipulation tool. Owada et al. [78] proposed several sketching user interface tools for region selection in volume data. Their tools are implemented as part of the Volume Catcher system [77]. Bürger et al. [14] proposed direct volume editing, a method for interactive volume editing on GPUs. They used 3D spherical brushes for intuitive coloring of particular structures in volumetric scalar fields. Abeysinghe and Ju [1] used 2D sketches to constrain skeletonization of intensity volumes. They tested their interactive tool on a range of biomedical data. To further facilitate selection and improve quality, Akers [4] incorporated a tablet screen into his segmentation system for neural pathways. Unfortunately, these methods were not used in any practical workflow to demonstrate their usability. Tools with interactive and intuitive volume segmentation were not available to end-users such as biologists. They would like a tool that could combine the effort-saving convenience of automatic segmentation algorithms and versatile user-guidance of manual segmentation.

2.5 Synthetic Brainbows

Randomness is an inherent character accompanying all natural processes. Researchers in biology have taken advantages of randomness. We are particularly interested in one recent technique in life science: Brainbow. In [63], Livet et al. described a series of strategies to randomly express fluorescent proteins in individual cells of mouse nervous system. They exploited the advantages of the widely used Cre/lox recombination system [11], which is able to turn on or off the expression of one or several different fluorescent proteins in a gene sequence. For different cells that are genetically modified to work with this technique, different combinations of the fluorescent proteins can occur. This is because the Cre/lox recombination system randomly chooses the gene expressions for recombination. The end result is that different cells, despite

the same type, are fluorescently stained with different colors. This technique is useful to visualize and distinguish detailed structures in the nervous system, where cells are packed and can be touching. However, the application of the Brainbow technique on a certain species of animals is limited because of complex transgenic manipulations. Disambiguation of cellular structures in single-channel datasets are commonly addressed by segmentation techniques (e.g., Cohen et al. [20]), which usually change the appearance of the original data and may be undesired for visualization purposes. Inspired by Brainbow, we would like to use computational techniques to randomly colorize confocal microscopy data processed with common antibody staining. By generating Brainbow-like results where different structures are distinctively colored, our proposed technique is an improvement to the visualization of single-channel confocal microscopy data.

Our Synthetic Brainbow technique leverages a process known as GPU framebuffer feedback loops. This process reads and writes the same framebuffer by multiple rendering or computing threads on GPU. If the output value of one pixel is dependent of its neighbors' values, it essentially creates race conditions among different threads. Without locking or synchronizing of the threads, the results become nondeterministic. Experienced graphics program developers avoid the nondeterministic behavior of GPU framebuffer feedback loops by framebuffer Ping-Pong, which is a technique using two framebuffers for reading and writing, thus synchronizing different threads. In fact, setting up a framebuffer feedback loop by binding the same framebuffer to a shader's (or computing kernel's) input and output is not considered an error by graphics hardware specifications. Developers are simply warned against doing so because the results are "undefined" [91]. However, considering that a framebuffer feedback loop is computationally more efficient by saving half the memory and using fewer context switches, we do believe it deserves a closer examination. In our research, we find that the nondeterministic behavior of given graphics hardware induced by asynchronism can be statistically tested and determined.

Applications of GPU framebuffer feedback loops are rare in previous work, due

to the fact that deterministic results are generally desired in computations with GPUs, such as filtering in image processing and equation solving in simulations. However, its theoretical development has long preceded the appearance of GPUs. An iterative computational model of GPUs is equivalent to a cellular automaton. The framebuffer can be thought as a grid of the cellular automaton with its pixels as the cells. The shader or computing kernel provides the rules for updating the states of the cells. The study of cellular automata dates back to the early history of computer science, including work of Ulam and Neumann [107]. Different types of cellular automata have been extensively studied through the later development of computer technology. Cellular automata have been proposed as computational models for simulations in physics [67] [86], material science [9] [94], and biology [42] [72]. They have also been extensively used in image segmentation algorithms [103] [47] [48] [49] [31] [62]. However, most research focused on synchronous cellular automata, where the state of every cell is updated together. Using framebuffer Ping-Pong in an iterative computational model is an example of a synchronous cellular automaton. In contrast, a framebuffer feedback loop should update individual cells independently, and the new state of a cell affects the calculation of states in neighboring cells, thus an asynchronous cellular automaton. Asynchronous cellular automata are generally less studied due to their nondeterministic behaviors. A lot of effort has been spent in recent research on asynchronous cellular automata to find efficient ways of computing deterministically without global synchronization. The research is mostly based on chaotic relaxation (Chazen and Miranker [17]), which described necessary and sufficient conditions for an asynchronous and chaotic process to converge. Baudet [8] presented a class of asynchronous iterative methods for solving a system of equations. Adachi et al. [2] presented an asynchronously updating cellular automaton that conducts computation without relying on a simulated global synchronization mechanism. Galilée et al. [30] proposed a joint algorithm-architecture for computing watershed segmentation. Their algorithm is programmed as a set of concurrent communicating iterative

programs that are efficiently mapped onto an asynchronous parallel architecture. Venkatasubramanian and Vuduc [102] described GPU implementations of Jacobi's iterative method for the 2D Poisson equation. Their implementations include a "wild" asynchronous example, which removed synchronization between iterations. They have shown that the "wild" asynchronous implementation on GPU has 1.2-2.5x speedups against best synchronous implementation, thanks to highly efficient memory bandwidth utilization. For GPU implementations of connected component labeling, Oliveira and Lotufo [73] discussed an ID merging method using asynchronous automata and presented an improved algorithm that included local and global merging stages. They also reported their method achieved 5-10x speedup in relation to Stephano-Bulgarelli's [25] serial algorithm. We regard the previously presented connected component labeling methods as primitive forms of more sophisticated colorization. Different from previous research, we leverage the randomness and use a computational stochastic process to simulate the results from a biological stochastic process: Brainbow.

2.6 Anatomical Atlases

An anatomical atlas provides a detailed map for medical and biological studies. The continuous efforts for making atlases of human anatomy date back to the work of many renowned anatomists: Vesalius, Leonardo da Vinci, William Hunter, and Henry Gray, whose creations are not only esteemed for their scientific value, but also appreciated aesthetically as masterpieces of art. Truly an arena where science meets art, anatomical atlases evolved as technologies advance in both fields: painting, printing, photography, microscopy, tomography, and certainly computer graphics. Whenever a novel technology emerges, our knowledge of anatomy is enriched with both exciting scientific findings and the increasingly detailed information presented in an atlas. Historically, an anatomical atlas has been a book of illustrations and text that systematically explains the anatomy of particular biological species. Naturally, the anatomy of humans has been the most studied. The most influential printed human anatomy atlases available today are

Gray's [34], Netter's [71], and Thieme's [32]. Interestingly, the production and appearance of printed atlases has changed with the development of technology. Henry Vandyke Carter, the illustrator of Henry Gray's anatomy book, drafted his illustrations in reverse on boxwood, which was then engraved for printing [83]. Frank Netter enjoyed the convenience brought by photography and modern printing. He chose a painting technique, gouache, which could better render the highlights and shadows to give his illustrations a more three-dimensional look. The illustrations of Thieme's Atlas of Anatomy were mostly hand drawn with Adobe Photoshop by Markus Voll and Karl Wesker. Though largely following the styles established by their predecessors, the use of digital media gives their illustrations finer details, smoother tonal gradations, and better transparency effects. All of these contribute to the clear representations of particular anatomical features. Printed atlases of other biological species are relatively scarce, and those in existence are usually decades old. For example, Greene's Anatomy of the Rat [35] was published in 1935 and is still used today as the definitive text for identification of anatomical structures in all rodents. In particular, Greene's atlas is used as *the* anatomy text for the laboratory mouse, one of the most important model organisms in biology and medicine. Anatomical atlases are crucial to understanding normal anatomy and identifying congenital abnormalities. For educational purposes, physical models are also sometimes built and used in addition to anatomy books. Physical models provide a unique three-dimensional model of anatomy. However, they are difficult to make, store, and maintain. Hardly can physical models achieve the level of detail required by scientific research, and thus they are rarely used in biological or medical research.

Computer graphics not only revolutionized the film industry, but also transformed anatomical atlases. Computer-generated atlases allow for a 3D, manipulable visualization of anatomy. Several 3D atlases have been generated for human anatomy, including Visible Body (www.visiblebody.com), Cyber-Anatomy (www.cyber-anatomy.com), and Zygote's anatomical model library (www.zygote.com). Since 3D models are commonly built by referencing 2D illustrations

of anatomy books, building 3D atlases is expensive and requires specialized personnel with experience in both digital modeling and anatomy. For biologists to build 3D anatomical models, it is most practical to use 3D scanned biological samples. Due to noise and limited resolution, anatomical atlases [40] that directly use volume renderings of scanned 3D volumetric data usually cannot achieve the clarity of polygon-based atlases, composed of objects modeled by representing their surfaces with polygons. There are several publications of polygon-based anatomical atlases for biological research (e.g., Ju [46] and DeLaurier [24]). However, easy-to-follow workflows and examples are unavailable for biologists to learn to make such atlases.

CHAPTER 3

A VISUALIZATION PIPELINE FOR CONFOCAL MICROSCOPY DATA

3.1 Confocal Data Formats as Inputs

Though different confocal microscope manufactures support different raw formats, confocal microscope users commonly use TIFF as both destination and exchange formats for storage, visualization, processing, and publication. TIFF files can be easily handled by standard image input/output libraries within most programming environments. However, these special features introduced by confocal microscopy datasets require us to have our own support for confocal microscopy file formats.

• Input precision. Depending on the model of the photodetector and Analog-Digital Converter (ADC) used with the microscope, the bit-depth of each confocal channel varies from 8-bit to 16-bit. TIFF is designed to work with an arbitrary number of bit-depth, but 8-bit and 16-bit are the most commonly seen. In fact, for 16-bit format, there are usually 12 or 14 significant bits. Rendering results will be considerably dark if we simply duplicate such data into graphics memory. To preserve input data precision, the actual bit-depth of an input dataset is first acquired by parsing the dataset and searching for the maximum intensity value. When the actual bit-depth of a dataset is 8-bit, data blocks from the original file are copied into graphics memory as 8-bit 3D textures; when the actual bit-depth is over 8-bit, data blocks are copied into graphics memory as 16-bit 3D textures. The actual bit-depth is used as a modulation factor to generate correct rendered brightness.

• Volumetric data. For confocal images, there are two competing methods

when using TIFF for volumetric data storage: a sequence of 2D image sections and a single file with multiple pages. Multipage TIFF files are easier to manage, since each file contains a single dataset. However, they may not be universally supported as 2D TIFF sections are. For example, Adobe Photoshop, which is commonly used by biologists, does not have native support for multipage TIFFs. To support both methods, the format importer in FluoRender will first decide which of these two methods is used. Then for a 2D section sequence, files in the same folder with similar names are enumerated, matched, and ordered to form the correct data sequence. On the other hand, reading a multipage TIFF file is rather straightforward. For large datasets, bricking is used. These datasets are usually the scans of a moving stage, where a specimen is scanned at different regions and these regions are later mosaiced to form a large dataset. The brick size is determined according to graphics cards' capabilities. As of the time of writing (ca 2012), both major gaming and professional graphics device manufactures, i.e., AMD and nVIDIA, provide mainstream products with 2GB-6GB of graphics memory. However, there is a discrepancy in their support of 3D textures: the maximum size supported by nVIDIA is 2048 pixels; most AMD's graphic cards support 8192 pixels at maximum and sometimes 16384 pixels for certain professional products. For most confocal data in practice, both are sufficient and no bricking is actually needed. In addition to pixel size of a confocal volume, another important parameter retrieved from TIFF metadata is the physical resolution, usually in micrometers, which describes the physical size of a voxel in X, Y, and Z directions. Notice that a confocal dataset is anisotropic in most cases. The resolution is crucial for a correct proportion when the dataset is rendered.

• Multichannel data. TIFF is also designed to support an arbitrary number of channels. However, saving multiple confocal channels using standard RGB color channels is the most common, since there are usually limited number of laser wavelengths and fluorescent tags. Sometimes, laser wavelength information is available in TIFF metadata. Such information is retrieved and used for assigning colors to different confocal channels automatically. When this information is not

available, we assign red, green, and blue to the first read channels by default.

• Time-sequence data. A time-sequence dataset is generated from continuous imaging of a living specimen. Despite the possibility of storing a complete time sequence with a single TIFF file, each time timepoint is usually stored separately. Confocal time sequences are sometimes referred as 4D or 5D data, because they are volumetric (3D), time-varying (the fourth dimension), and multichannel (the fifth dimension). Rendering time-sequence in a timely manner is most important for users, since many biological phenomena, such as mitosis, can be visually detected if an image sequences is played smoothly. We try to minimize the playback latency with three means. First, we reduce reading and processing time for each timepoint. Only necessary operations are performed when reading one timepoint, such as retrieving data information and data block copying, since most data processing and enhancement are deferred to later in our visualization pipeline. Second, information for each timepoint is prefetched and cached, so that it is always immediately available when data for any timepoint are requested. Third, the actual data for each timepoint are read from disk when they are requested during first-time playback, and then they are cached in system main memory. Although this design choice makes the speed of first-time playback dependent on disk speed, which can be slow for low-end systems, the advantage is that visualization of a time-sequence data is instantly available only after the first timepoint is loaded. The speed of the second and later playbacks can still catch up. We tested and compared the visualization speeds of FluoRender and two other tools commonly used for time-sequence confocal data visualization, i.e., Volocity and Imaris. The test results are shown in Figure 3.1. The test was comprised of four subtests. Dataset loading tested the time of loading a 3.43GB dataset, which contains 210 timepoints of a 12-hour continuous imaging. Since FluoRender gathers data information of all timepoints and only reads the actual data block of the first timepoint at initial loading, the latency is negligible. Total operation time is the duration between the application launch to when the first timepoint is visualized. It seems that both Volocity and Imaris preprocess the dataset, which

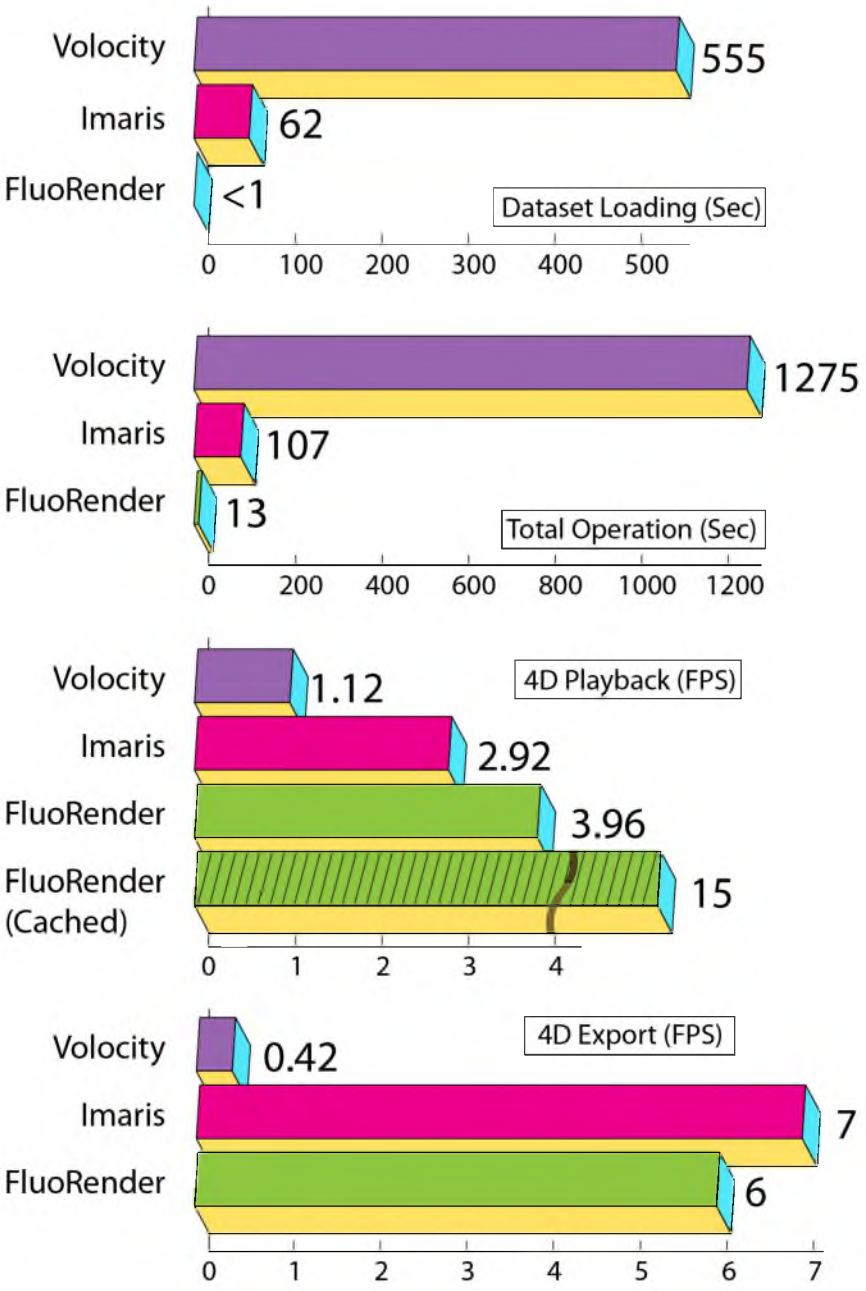

Figure 3.1. Speed comparisons. We tested all speeds on the same PC with Intel Core i7 3.2GHz, 12GB memory, single 7200 RPM SATA disk, nVIDIA GTX280 and Microsoft Windows XP 64bit. The dataset is a two-channel 4D confocal dataset with 210 frames, which occupies 3.43GB on disk. Volocity is 64bit at version 5.1.0. Imaris is 64bit at version 6.3.0. FluoRender is 64bit at version 2.9.0

causes considerable delay before any result can be visualized. The difference between 4D playback and 4D export is that 4D export saves the result, typically to a 2D image sequence. We calculated the speed by dividing the total number of timepoint (210) with playback time. FluoRender caches data using available system memory during first-time playback, and the speed is considerably faster for later playback when data size is smaller than the system memory. Volocity and Imaris both require loading the entire dataset into system memory before playback, which makes them impossible to use when the size of a time sequence is larger than system memory. This seems to be no problem for FluoRender, as we learned from our neurobiologist users that FluoRender worked stably with a 50GB time sequence dataset on a common PC desktop.

In addition to the TIFF format, we also support two confocal manufacturers' raw formats, i.e., OIB and OIF from Olympus, and LSM from Zeiss. All formats have native support for volumetric, multichannel and time-sequence data. Since these formats are either wrappers around standard TIFF (OIB format uses Microsoft's structured storage to save standard 2D TIFF files), or modified TIFF (LSM format extends the TIFF specifications to support features such as larger-than-4-GB data), it becomes straightforward to support these customized formats by modifying our existing TIFF reader. Thus, the same reading/caching strategy as for standard TIFF-based confocal data is used for the manufacture-specific formats. However, the most significant difference is how confocal information (data size, resolution, laser wavelengths, etc.) is stored. So we designed our readers according to manufacturers' format specifications, which can be obtained from the above manufacturers upon request.

3.2 Intuitive and Efficient Transfer Function Manipulations

In FluoRender, 2D transfer functions [53] are used for setting rendering properties of volume data, as their boundary extracting capability can render fine structures from confocal data. We found, however, that biologist users prefer intuitiveness and efficiency to complicated transfer function widgets and settings.

With this in mind, we chose a family of the 2D transfer functions that best suits confocal data structure extraction, while the parameters for fine-tuning the shapes of the transfer function are chosen and named for better operability. The shape of the 2D transfer function, as well as the parameters, are illustrated in Figure 3.2.

• Boundary extraction. It controls the cut-off value of gradient magnitude. Setting a higher value can isolate better-defined boundaries in the volume data. Figure 3.3(c) shows that spreading of nuclei is seen in a combined rendering with other channels. By increasing the boundary extraction value, only the voxels defining nucleus boundaries are rendered. Combined with transparency adjustment, both the underlying channels and the spreading of nuclei are seen, which is not possible by adjusting transparency solely (Figure 3.3(b)).

• Saturation point. It offsets the intensity turnpoint in the 2D transfer function. Low intensity signals are enhanced when its value is lowered. Figure 3.3(e) and (f) show that the continuity of neuron fibers is recovered after adjusting this parameter.

• Low and high thresholds. They set the low and high cut-off values of scalar intensity. These values are useful for noise suppression. Figure 3.3(g) and (h) show an example before and after the threshold values are adjusted; noisy data are eliminated after adjusting the low threshold value.

• Gamma. It is a nonlinear falloff control of low intensity signals. It controls how values off the intensity peak are attenuated by adjusting the exponent of the intensity values. Gamma is adjusted to get a better contrast of the output renderings.

For a multichannel volume dataset, transfer function for each channel can be adjusted independently. FluoRender lets users interact with a limited set of parameters, with each parameter adjusted by either linked slider or numerical entry. The corresponding parameter settings in the user interface are listed in Figure 3.4. By avoiding complicated widgets or the jargon of transfer function settings, the provided interface is more intuitive for biologists to use and can quickly obtain the desired visualization results. In addition, default settings

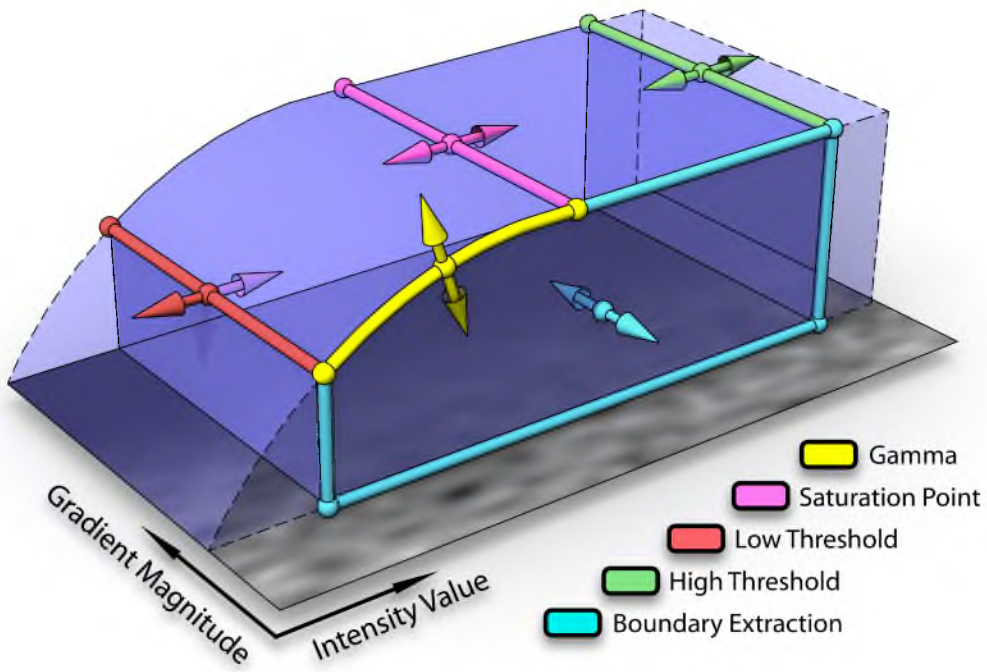

Figure 3.2. An illustation of the 2D volume transfer function in FluoRender. The colored arrows indicate the possible adjustments for the parameters, which are: gamma, saturation point, boundary extraction, low threshold, and high threshold.

coming with FluoRender installation packages are determined by our collaborating neurobiologists through their general workflows. All the transfer function parameters have default values, and they are automatically applied to loaded data. Users can also generate their own default settings after they become familiar with the settings.

It is a common practice that the volume transfer function is rasterized as a texture, and updated every time the parameters change. Two problems may occur if the texture transfer function is used. First, there is quantization error, especially when the transfer function is nonlinear, since texture lookup only uses linear interpolation. Second, it is impractical to build a texture transfer function for 16-bit data due to texture size limitations. Since our 2D transfer function has only five customized parameters for confocal visualization, we pass the parameters into the shader computing the volume rendering result, and evaluate

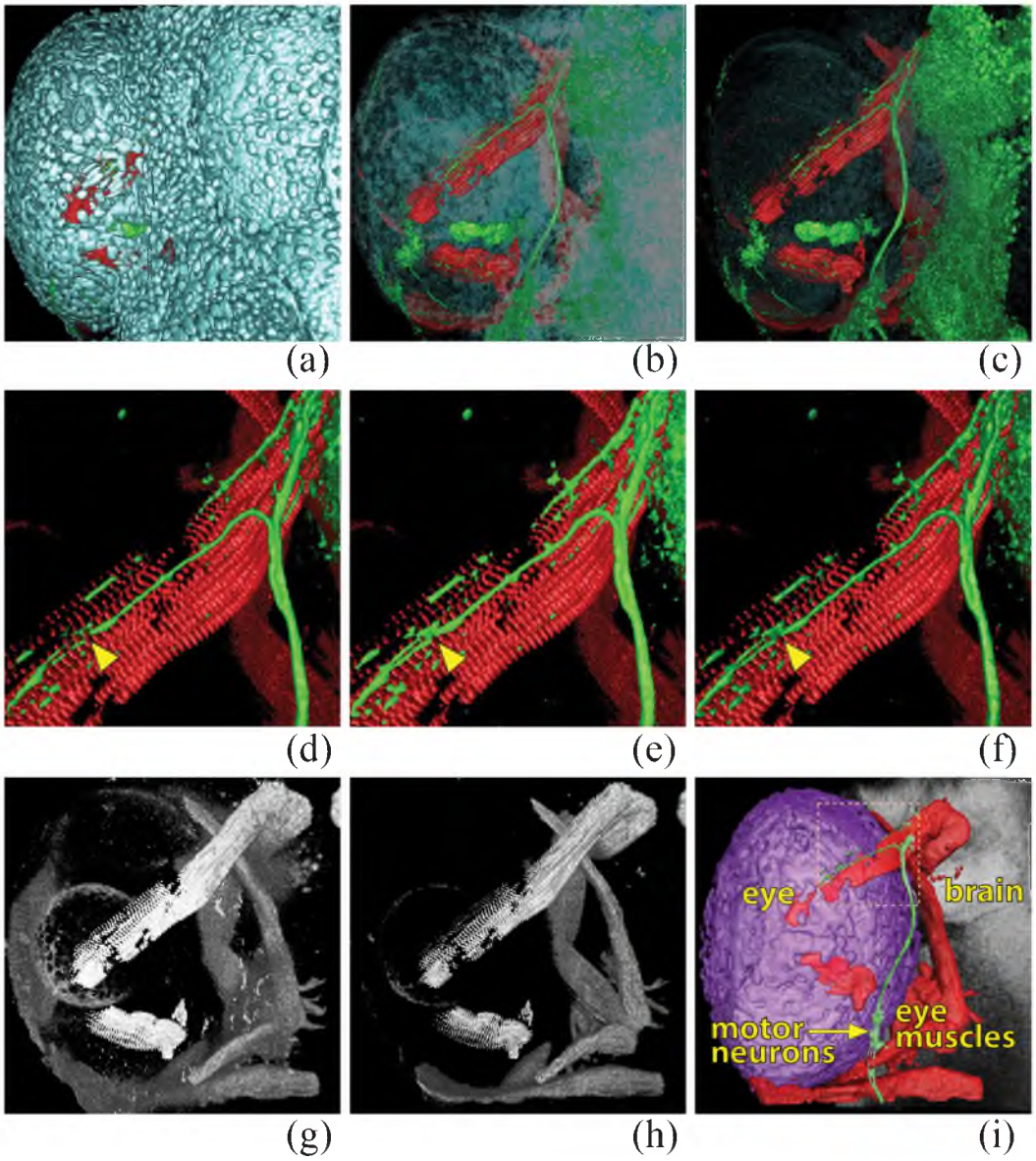

Figure 3.3. Results when our 2D transfer function is adjusted for a zebrafish head dataset. (a) The channel of nuclei (cyan) obstructs other channels. (b) Increasing the transparency may be helpful, but it makes the rendering obscure, and underlying channels are still partially occluded. (c) Increasing the boundary extraction value can better show the spreading of the cells and underlying channels. (d) The motor neurons (green) projecting to the eye muscles appear artifactually disconnected, indicated by a yellow arrowhead. (e) Adjusting the saturation point reveals that motor neuron fibers are in fact connected. (f) Shading helps better define the shape. (g) Low scalar intensity noise is present in the eye muscle channel (red channel in others). (h) Increasing the low threshold suppresses the noise. (i) A map of the regions analyzed.

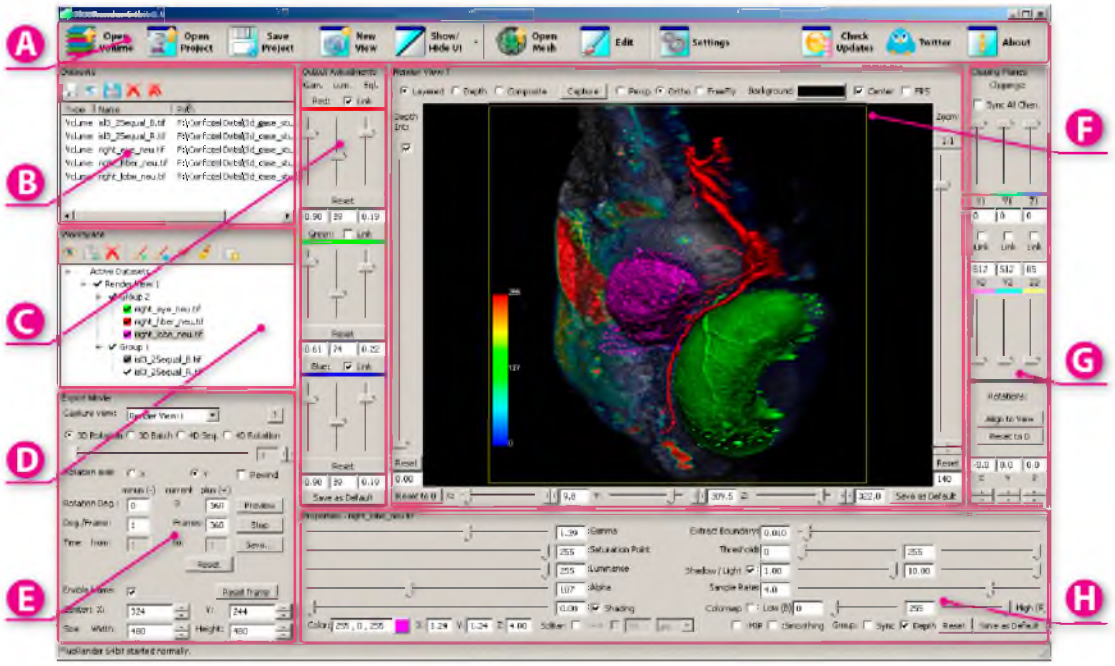

Figure 3.4. FluoRender user interface. A - Toolbar; B - List of loaded datasets; C - Tone-mapping adjustment; D - Tree layout of current active datasets; E - Movie export settings; F - Render viewport; G - Clipping plane controls; H - Volume data property settings.

the volume transfer function on the fly. The real-time evaluation of the transfer function ensures the rendering quality of the low-intensity signals in confocal data. Figure 3.5 compares the resulting difference between prequantized and real-time evaluation of the transfer function. While many other tools either do not have the flexibility of changing volume transfer function, or provide too many parameters and widgets that make evaluation on the fly impossible, FluoRender lets its users quickly adjust the volume transfer function for finely-detailed visualizations.

The volume rendering results of FluoRender always output to 32-bit floating-point framebuffers, and then all the 2D image space methods discussed subsequently are calculated with 32-bit precision. This is essential to our tool for high-precision adjustments.

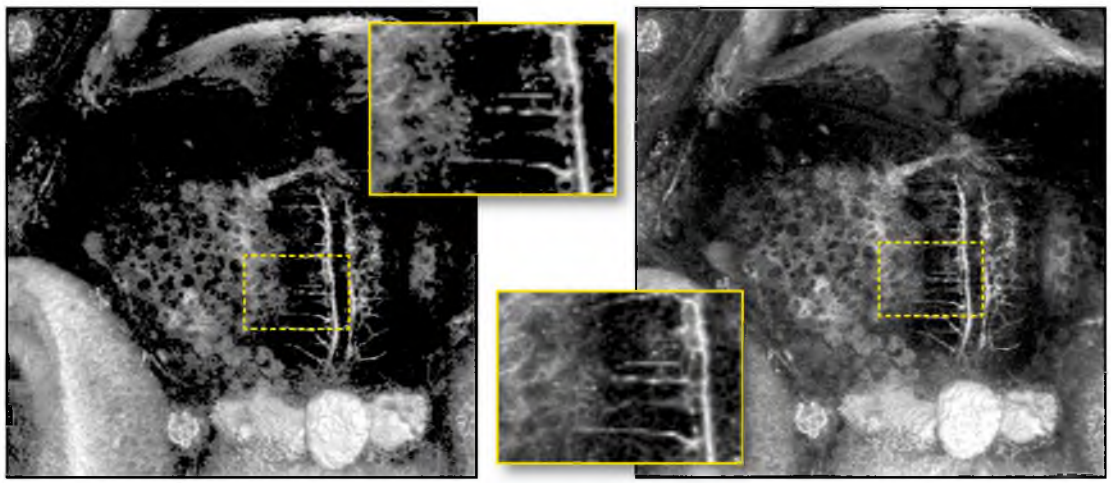

Figure 3.5. Results from prequantized (a) and on the fly (b) evaluation of the transfer function. The two results are generated with the same transfer function settings. The prequantized transfer function clips many details in the low intensity regions, which are preserved with on the fly evaluation of the transfer function. The dataset shows tectal neurons of a 5-day-postfertilization (5dpf) zebrafish.

3.3 Multiple Render Modes for Multichannel Data

For multichannel confocal microscopy data, qualitative analysis usually requires visualizing the spatial relationship between data from different channels. When combined together, however, data from different channels often interfere with each other, and details of interest from one channel can be occluded. We provide three render modes suggested by our collaborating biologists for multichannel volume data. The three render modes are: Depth mode, Composite mode, and Layered mode. Figure 3.6 compares the results of same three-channel dataset with different modes.

• Depth mode. When implemented with a slice-based volume renderer, multichannel volume data are blended first for each polygon slice and then the slices are blended together. This is the correct way to show the spatial relationships between channels, and most visualization tools that support multichannel datasets use this mode. However, sometimes fine structures from one channel are covered by voxels from other channels with lower depth values. Lowering the transparency

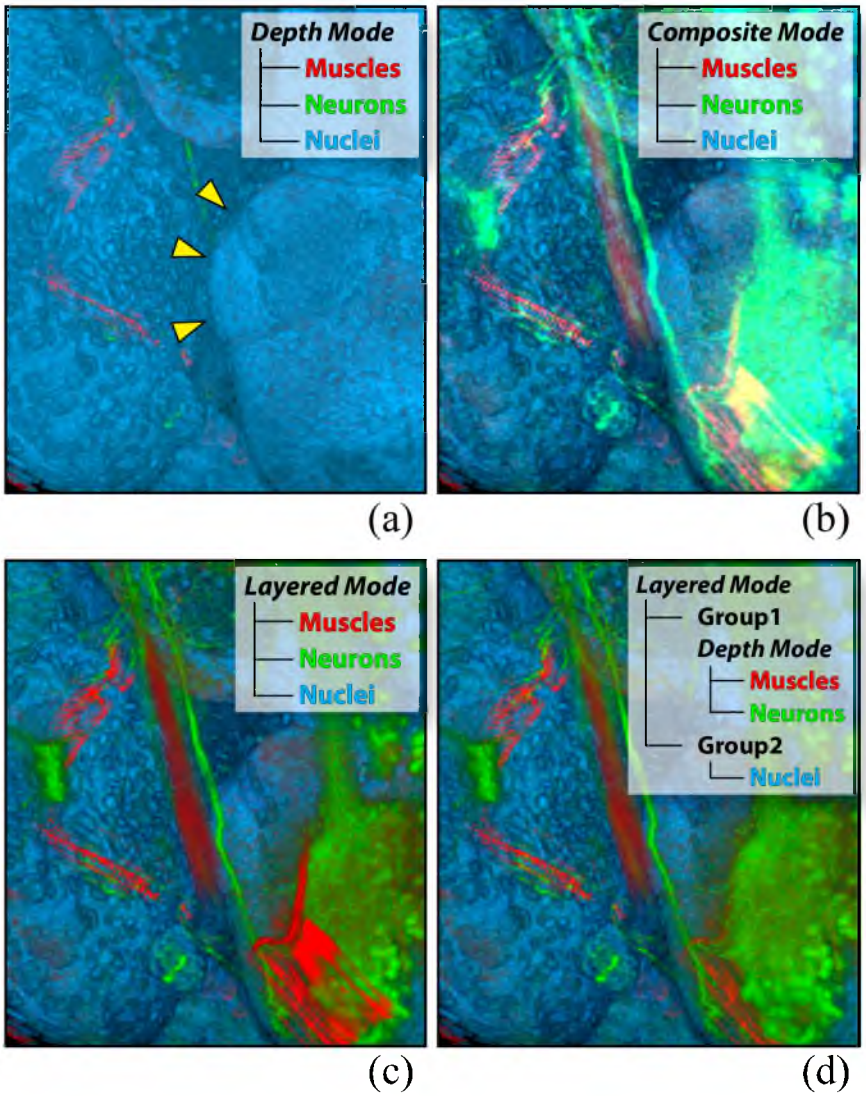

Figure 3.6. A confocal dataset of a 5dpf (days postfertilization) zebrafish embryo has three channels: eye muscles (red), neurons (green), and nuclei (blue). (a) The channels are combined with 3D compositing. The muscle and neuron channels are barely seen. Yellow arrowheads indicate the boundary of the brain, which is on the right side of the eye when visualized as in the figure. (b) The channels are composited with 2D addition. Highlight details are over-saturated, due to the additive compositing. (c) The channels are composited with 2D layering. Details of the muscle and neuron channels are visualized, but the spatial order of the two is incorrect. (d) The muscle channel and the neuron channel are grouped and combined with 3D compositing, which renders their spatial relationship correctly; the nuclei channel is in a separate group. The two groups are composited with 2D layering. The nuclei channel is a context layer, showing the boundary between the brain and the eye.

of the obstructing data can reveal the deeper structures, but usually the details of the obstructing data are lost (Figure 3.6(a)).

• Composite mode. Each channel of a multichannel volume data is first rendered into a texture, and the textures are composed into the final rendering with color component addition. As shown in Figure 3.6(b), information from all channels can be seen at the same time, as long as distinguishable colors are used. As it is not necessary to increase the transparency of the occluding channels, the renderings of all channels are bright and full of details. As most datasets in confocal research have three channels or less, it is most effective to set colors as pure red, green, and blue. Shading effect calculation is clamped to single color components if data channels are set to pure red, green, and blue. Therefore, the original data channel information can still be extracted from the exported images of this mode, by separating the color channels. This is important for further processing and publishing.

• Layered mode. Similar to layers in 2D painting software, the volume data are layered on top of one another. Each channel of a multichannel volume data is first rendered into a 2D texture, and then the 2D textures are rendered in the order specified by users. In this mode, the top layer data cover the lower ones. This does not respect the relative depth relationships within the data, especially during user interaction. Visualization experts did not expect this mode to be effective. Surprisingly, biologists often prefer this mode since it can better show fine inner structures, such as neuron fibers, when placed in the top layer (Figure 3.6(c)).

Compositing all channels with a single render mode may not be sufficient, especially when there are many confocal channels, including derived data from segmentation and analysis. Users want to group certain channels and combine specific channel groups differently with 2D or 3D compositing. Our collaborating biologists resorted to 2D image processing packages for group compositing, since early versions of FluoRender were not capable of rendering groups with different render modes. In Figure 3.6, (a), (b), and (c) are the three basic render modes. The dilemma for the user is which to choose in order to visualize the correct spatial

relationship between the neuron and muscle channels, but leave the nuclei channel as the context. We extended the idea of render modes by simply organizing data channels into groups. A group contains an arbitrary number of channels and has an independent render mode for combining its channels. Different groups are again combined with a render mode. Figure 3.6(d) shows the result that biologists were pleased with: both the muscle and neuron channels are visualized, with the correct spatial relationship, and they have a clear context.

Groups also facilitate the parameter adjustment of the confocal channels. A user can group certain confocal channels, and set the parameters of the channels within the group to synchronize. Changes to the parameters of one channel are then automatically propagated to other channels of the group.

Biologists may also find features they need in each mode. Joint views of different render modes can allow even improved data comprehension. We provide an interface to allow biologists to switch between the render modes quickly, and multiple viewports can be set for different render modes, which can be operated separately, or synchronized to the same viewing direction. Multiviews are indispensable when comparing different datasets in the qualitative analysis workflow. Datasets from replicate samples or from mutants and wildtypes are visualized and compared in different views. Like the transfer function settings, users can set the views quickly and accurately, or let FluoRender remember the default settings for later comparison.

3.4 Embedding Polygon Data for Region Definition

Incorporation of biologically meaningful boundaries can greatly aid interpretation of confocal data. Unfortunately, boundaries often cannot be reconstructed simply by setting transfer functions. Polygon data become important means for region definition, because they can be generated from user-defined regions inside volumetric data. We first segment a given confocal data. Then, the segmented regions are converted to polygon data using the marching-cubes algorithm [64]. The polygon data can be further smoothed for better illustrative

purposes. For details about segmentation and polygon data processing, see Chapter 4 and Chapter 6. For some applications such as crude region definition or volume culling, simple polygon geometries can be generated on the fly, including cubes, spheres, and cylinders. These geometries cane be manually translated, rotated, and scaled to match specific structures in the confocal data. This is less time-consuming than manual segmentation, but is still sufficient for many cases in qualitative analysis where precision is not a major concern.

The difficulty of using polygon data with volume rendering arises when we want to render them with adjustable transparency. This is because we want to observe the volumetric data inside of the regions defined by the polygon data. In general, correct rendering order in space is expected when semitransparent objects are rendered. We use the depth peeling [28] algorithm, because it is a robust solution to the ordering problem when multiple transparent objects as well as volume data are rendered. Our implementation improves the original depth peeling algorithm by cutting volume data spatially and rendering both polygonal and volumetric pieces in sequential order. The algorithm is graphically presented in Figure 3.7.

In theory, the number of depth peeling layers can be determined automatically by querying the results from each peeling iteration. However, we provide a user interface to let users adjust the number layers for depth peeling. This is because the number of peeling layers has a great impact on performance. In many applications of qualitative analysis, one peeling layer can achieve a satisfactory result while maintaining high interactivity. With a higher peeling layer setting, better accuracy can be achieved, allowing better understanding of how the volume data and the polygon-defined regions are spatially related. For most complex geometries resulting from confocal data segmentation, we found four layers enough for sufficient accuracy. Figure 3.8 compares the difference between the depth peeling settings. The examples show how the positions of neurons relative to the eye and central brain can be better perceived.

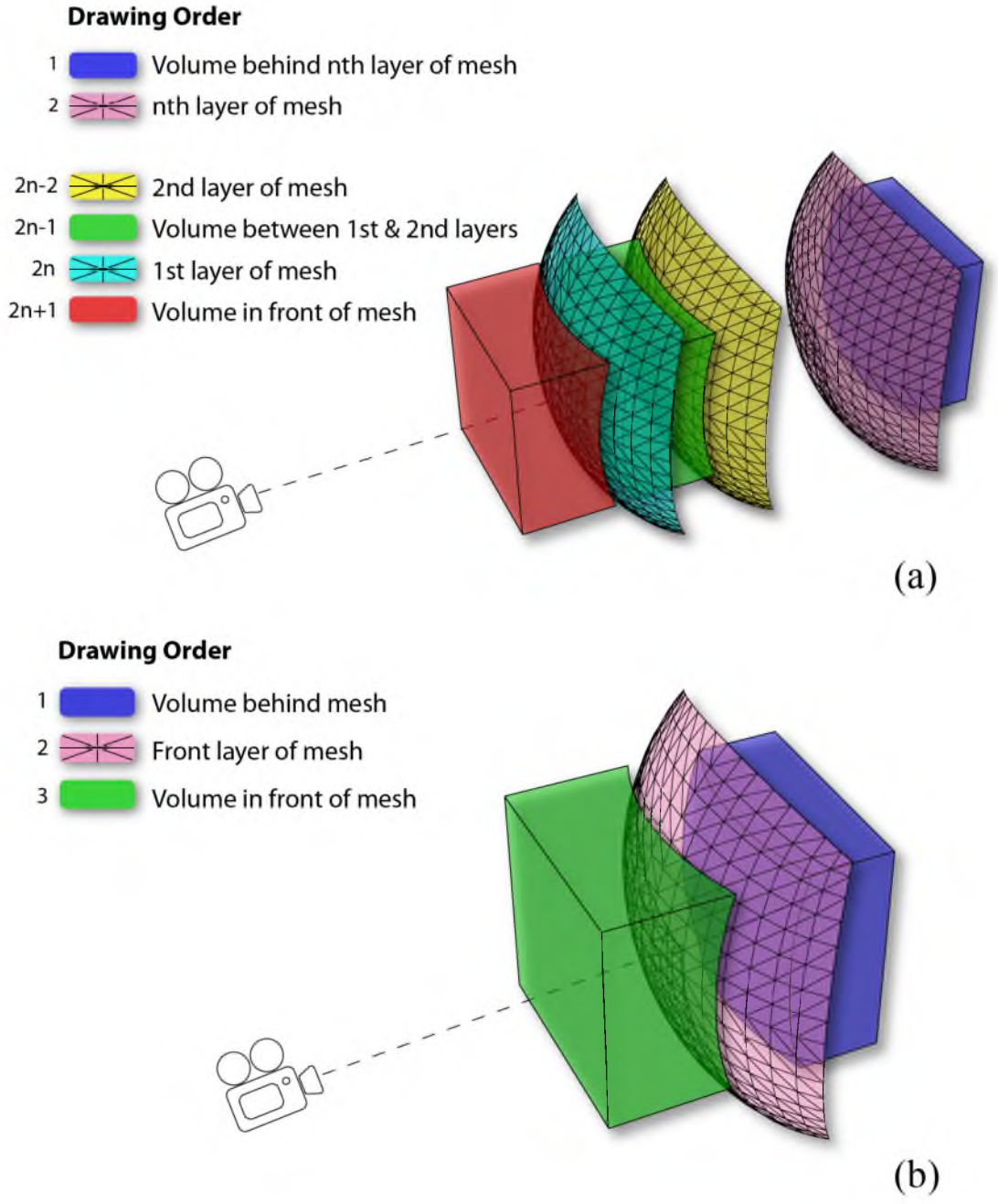

Figure 3.7. The depth peeling algorithm used in FluoRender. (a) When there are n depth peeling layers, volume data are separated by these layers and rendered with a correct order. (b) When there is only one depth peeling layer, the rendering is much simplified but usually still generates acceptable results.

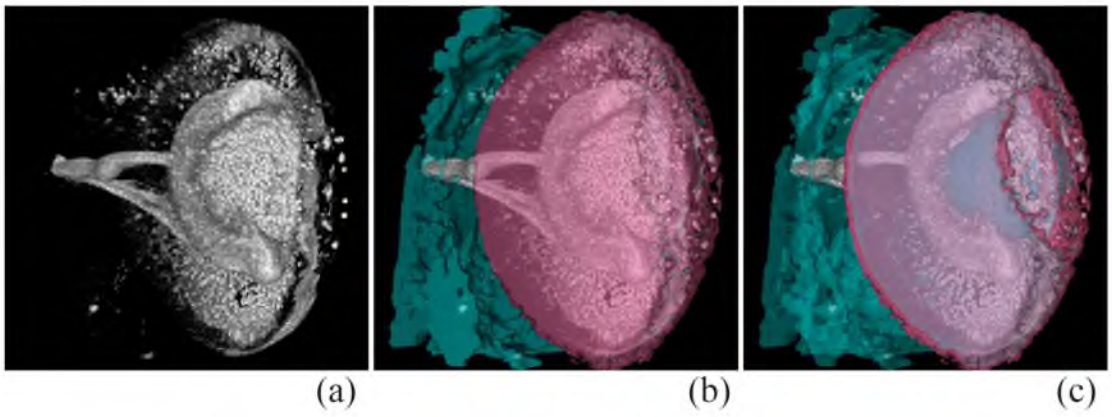

Figure 3.8. Depth peeling results. (a) Ventral view of the volume data showing retinal ganglion cells connecting between the eye and the brain. (b) Polygon data added, separating volume data into eye (magenta) and brain (cyan); depth peeling layers set to one. (c) Same data, depth peeling layers set to four. Arrowheads point to two branches of visual neuron fibers. With more depth peeling layers, it is clear that the lower branch is located deeper behind the eye region, which is not apparent in either (a) or (b). (Dataset: Zebrafish head)

3.5 2D Tone Mapping

2D tone-mapping operators can be found in many image processing packages but are absent from confocal visualization tools and most volume visualization tools in general. When only the volume transfer function is adjusted, biologist users sometimes found it difficult to achieve both satisfactory brightness and details for volume rendering outputs. This can be explained with the volume rendering integral. Consider the commonly used emission-absorption model, where the resulting intensity is calculated as [69]:

$$I(D) = \int_{s_0}^{D} q(s) e^{-\int_s^D \kappa(t)\mathrm{d}t} \mathrm{d}s \tag{3.1}$$

In Equation 3.1, q is the emission term, and κ is the absorption term. As shown in the two attached plots of Figure 3.9(a) and (b), when we apply a monotonic adjustment (for example, gamma correction, which preserves the order of its input intensity values so that they are globally darkened or brightened) $f(q,p)$ with a parameter p on q (or κ), the output intensity ($I(D)$) is generally not changing

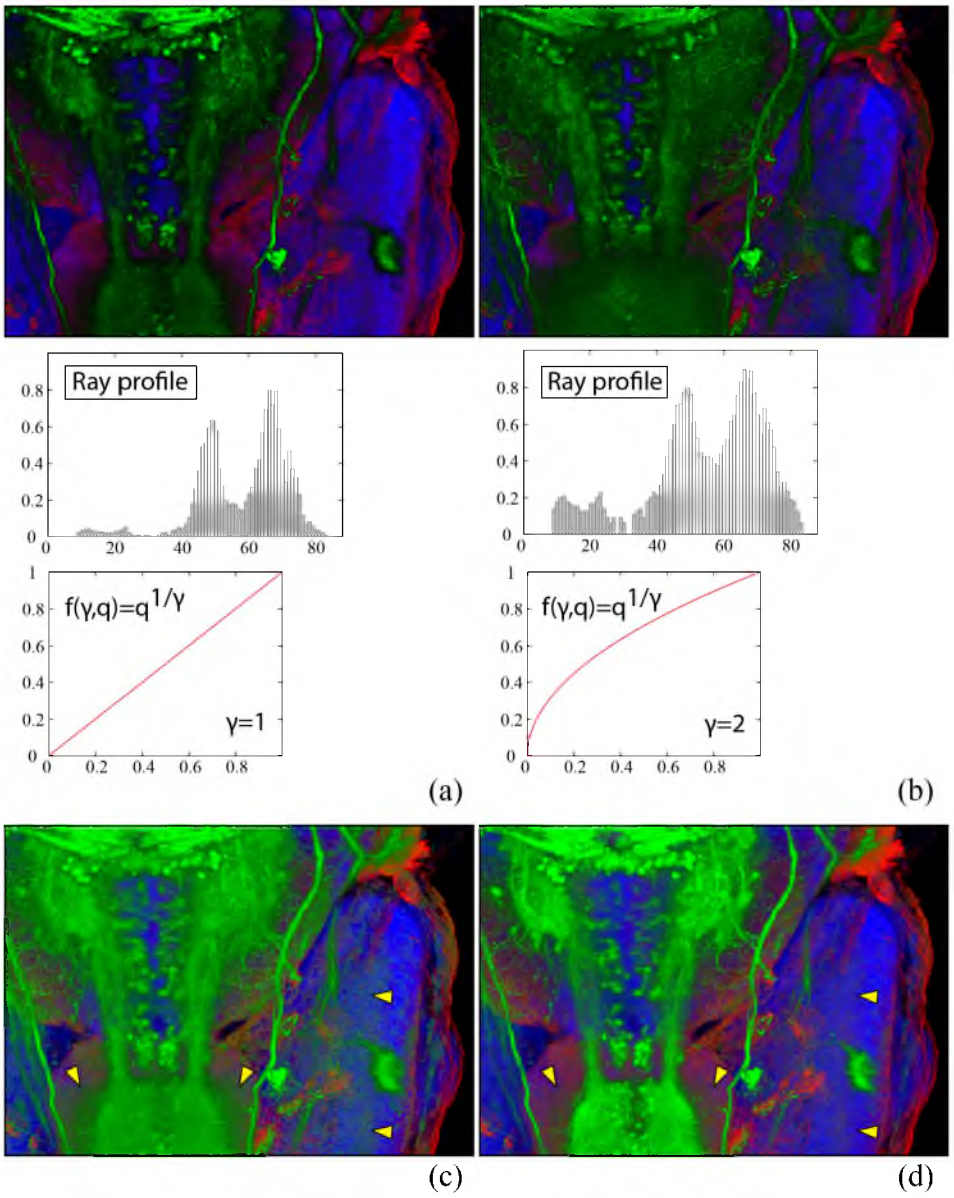

Figure 3.9. The volume transfer function nonlinear falloff and the 2D image space gamma adjustment work well together. The confocal dataset shows three channels of a 5dpf zebrafish embryo: eye muscles (red), neurons (green), and nuclei (blue). (a) The initial rendering without any adjustment. (b) The result when the transfer function gamma is increased. The rendering does not become brighter as many users may expect. The attached plots show the change of one ray profile after the transfer function gamma is increased. (c) The result when the gamma as 2D adjustment is increased. The result is brightened, but noise becomes prominent in the regions indicated by yellow arrowheads. (d) The satisfactory result achieved by decreasing transfer function gamma and increasing gamma as 2D adjustment. Neuron fibers are visualized clearly with less noise.

monotonically with p, due to the complexity of q (or κ) along the integration path (ray profile). The adjustment is usually embedded within volume transfer functions, and this nonmonotonic relationship between parameter and result makes it difficult for users to adjust for desired brightness. Here we apply tone mapping with monotonic adjustments on volume rendering outputs, which is intuitive to use for adjusting brightness.

The general definition of tone mapping is the mapping of one set of colors to another, but the term is mostly used with high dynamic range images (HDRI) (Reinhard et al. [81]), where the meaning narrows to compression of the high dynamic range of light information to a lower dynamic range. Since confocal data are acquired up to 16 bits per channel at present and strictly speaking are not HDR, we use the term tone mapping in this chapter in respect of the general definition. However, the objectives of HDR tone mapping still apply to confocal data visualization, i.e., rendering all possible tone ranges at the same time and preserving the details with local contrast in order to obtain a natural look. We implemented the following three tone-mapping operators, and made certain customizations specifically for confocal data.

Gamma correction is the most-used nonlinear operator in image processing. We follow convention and calculate the output color C_{out} with Equation 3.2:

$$C_{out} = C_{in}^{\frac{1}{gamma}} \tag{3.2}$$

The nonlinear adjustment of the low intensity falloff in our 2D transfer function is essentially the same gamma correction embedded within the transfer function. However, its actual influence on brightness is quite different from applying it in 2D: increasing the transfer function gamma enhances details for low intensity voxels (Figure 3.9(b)), which usually makes the rendering result less bright, and vice versa. The transfer function gamma is a parameter biologist users frequently use to either enhance or suppress low intensity signals in confocal data. However, the brightness of the results cannot be adjusted the same as one would expect

from gamma correction (Figure 3.9(c)). By adding gamma correction in 2D as an independent parameter of the transfer function gamma, biologists can adjust both details and brightness easily. For example, in Figure 3.9(d), the volume transfer function gamma is decreased to suppress noise signals, and the gamma as a 2D adjustment is increased to reveal the fine details of the neuron fibers.

Luminance is usually called exposure in photo-editing tools. It is a scalar multiplier on the input color, which is used to brighten/darken the overall rendering and expand/compress the contrast linearly. In order for the user to adjust the luminance intuitively, we customize its parameter L by mapping it to the actual factor with a piecewise function $f(L)$.

$$\begin{aligned} C_{out} &= C_{in} \cdot f(L) \\ f(L) &= \begin{cases} L & L \leq 1 \\ \frac{1}{2-L} & otherwise \end{cases} \end{aligned} \tag{3.3}$$

Figure 3.10 shows $f(L)$ in both linear and logarithmic scale plots. The function is pieced together from a linear function and a nonlinear curve, and is $C1$ smooth. The user-adjustable parameter L has range $[0,2)$. It darkens the result by compressing the dynamic range within $[0,1)$, and brightens the result by expanding the dynamic range within $(1,2)$. In the logarithmic scale plot, the curve is antisymmetric at the center point $(1,1)$, so in addition to a monotonic adjustment, our luminance operator gives an intuitive feel that the output is equally brightened or darkened when L is increased or decreased.

Scale-space equalization is a local tone-mapping operator that equalizes the uneven brightness and enhancing the fine details of confocal microscopy data. Using levels-of-detail with the scale space for tone mapping can be found in the work of Jobson et al. [45]. Instead of using logarithmic mappings for dynamic range compression, which is widely used in HDRI processing, we divide the input color (C_{in}) by the scale space color ($\overline{C_i}$), which is an average calculated by low-pass filtering. Thus, the input color is equalized at a series of detail levels, hence the name scale-space equalization. The output color of this operator is

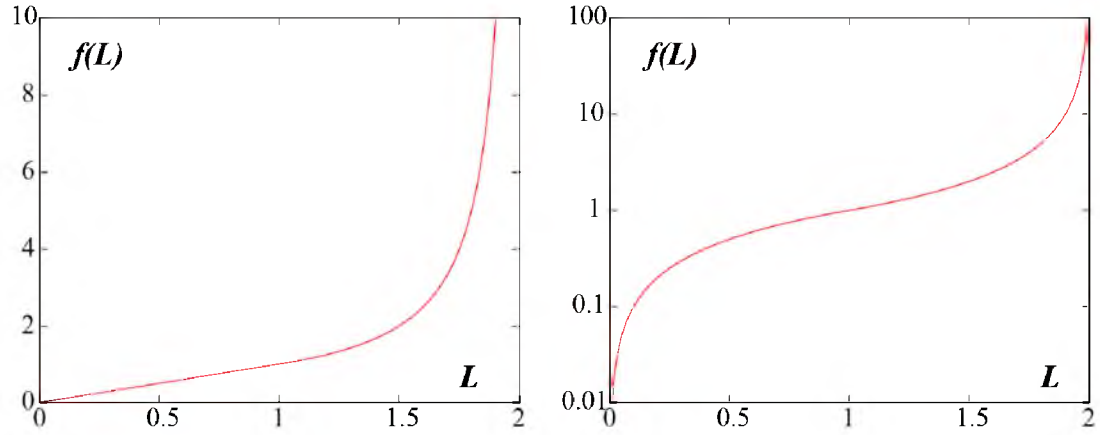

Figure 3.10. The mapping of the user-adjustable parameter L and the scaling factor $f(L)$, in linear scale (left) and logarithmic scale (right) plots.

calculated by a weighted sum of the equalized colors and then blended with the input color, as in Equation 3.4:

$$
\begin{aligned}
C_{out} &= (1-t)\cdot C_{in} + t\cdot C_{Eq} \\
C_{Eq} &= \sum_{1}^{N} v_i \cdot C_i \\
C_i &= \frac{C_{in}}{\overline{C_i}}
\end{aligned}
\tag{3.4}
$$

In Equation 3.4, v_i is a set of weighting factors, which are empirically determined by experimenting with typical confocal datasets. A plot of the v_i we use in FluoRender is shown in Figure 3.11. The only parameter exposed to the end-user is the blending factor t, which linearly blends the equalized color with the input color. This linear blending, which is missing even in most HDRI processing software, ensures a monotonic change to brightness as previously. Figure 3.11 illustrates the equalizing process. It also shows the results when the blending factor changes. The originally dark rendering of the confocal dataset is brightened, yet the fine details are still clearly visualized. For noisy confocal data, increasing the blending factor also enhances high frequency noise. Thus, noise removal through pre- or postprocessing is usually desired.

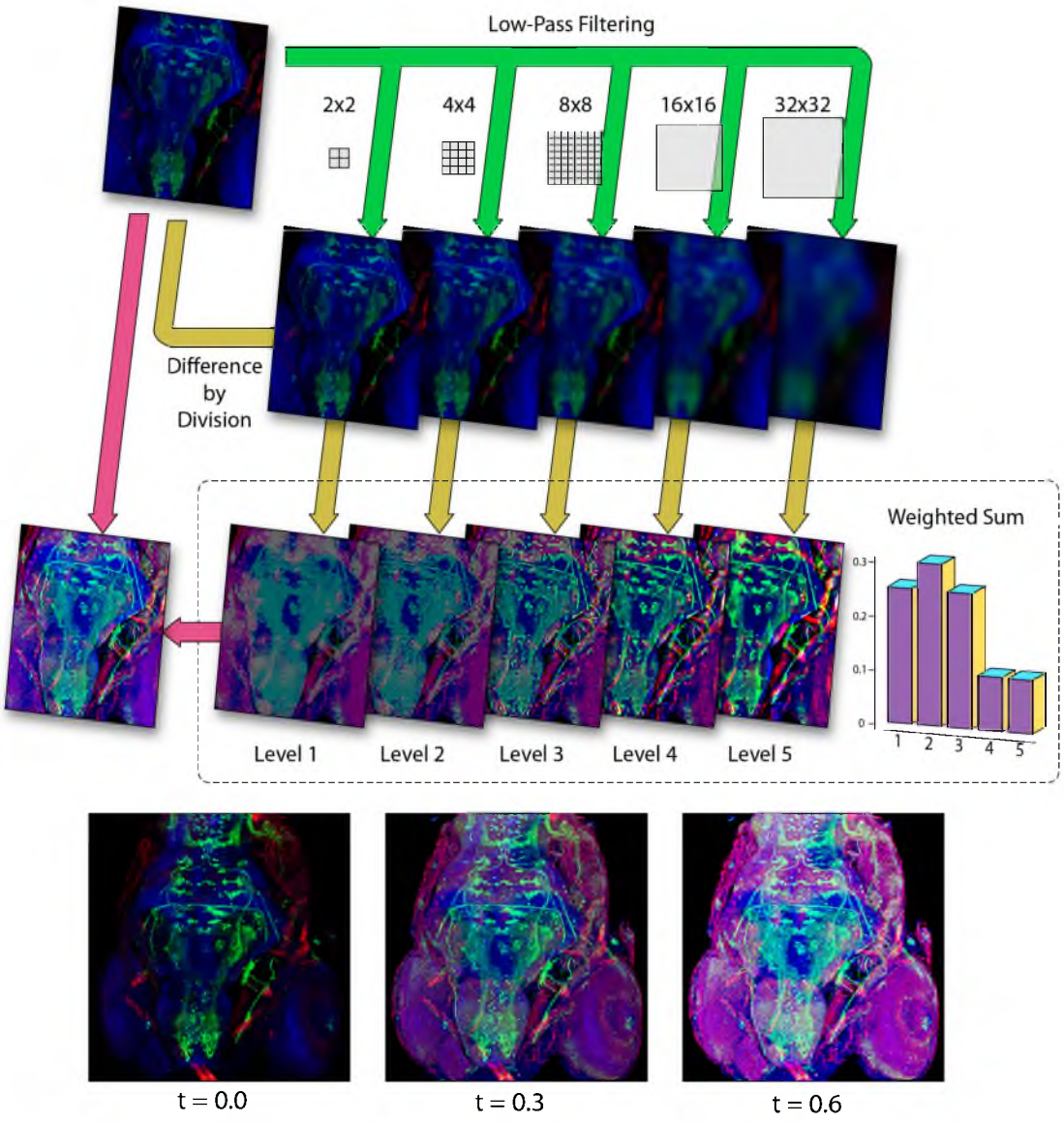

Figure 3.11. The scale-space equalization process. The example dataset has three channels of stained muscles, neurons, and nuclei of the zebrafish head.

The importance of scale-space equalization for confocal visualization is normalizing brightness – along the Z-axis for 3D channels and through time for 4D sequences. Figure 3.12 compares the results before and after 2D tone mappings applied to a two-channel confocal dataset. The dataset is the nuclei (magenta channel in Figure 3.12(a)) and neurons (green channel in Figure 3.12(a)) of a 5dpf zebrafish head. Biologist users want to study the shape and spatial relationship of the neurons in the region between zebrafish eye and brain. Figure 3.12(a) (dorsal view) and (b) (lateral view) show the volume rendering result with no 2D tone mapping applied and volume transfer function set to a linear ramp. They also represent the results from most other confocal visualization tools when the dataset is loaded. Though the general shapes of its major structures can be visualized, such as the eyes, the brain, and the tectum, many details, especially those in the neuron channel, are either occluded or not clearly seen. The lateral view of Figure 3.12(b) shows a common problem for confocal data: the brightness is decreasing along the Z-axis (from dorsal to ventral for this dataset). This is the direction in which the laser beam travels; due to scattering and tissue occlusion, signals become weaker as the scanning goes deeper along this direction. Since the brightness decrease is sample dependent, a simple calibration of the microscope cannot correct it. Figure 3.12(c) and (d) show the results from the same view directions, however rendered with FluoRender's default transfer function settings and 2D tone mappings applied. The scale-space equalization operator brightens the signals deeper along the Z-axis. In Figure 3.12(c), the brightness is even, yet the details of the neural structures are enhanced as well.

3.6 MIP Enhancement with 2D Color Mapping and Overlays

A confocal channel is a scalar volumetric dataset, whose values represent the fluorescent intensities, which in turn measure amounts of biological expression. Biologists often want to assess the amount of gene/protein expression with better quantification than just rendering intensities. Furthermore, since high-intensity values represent strong biological expression, it is important to visualize them over

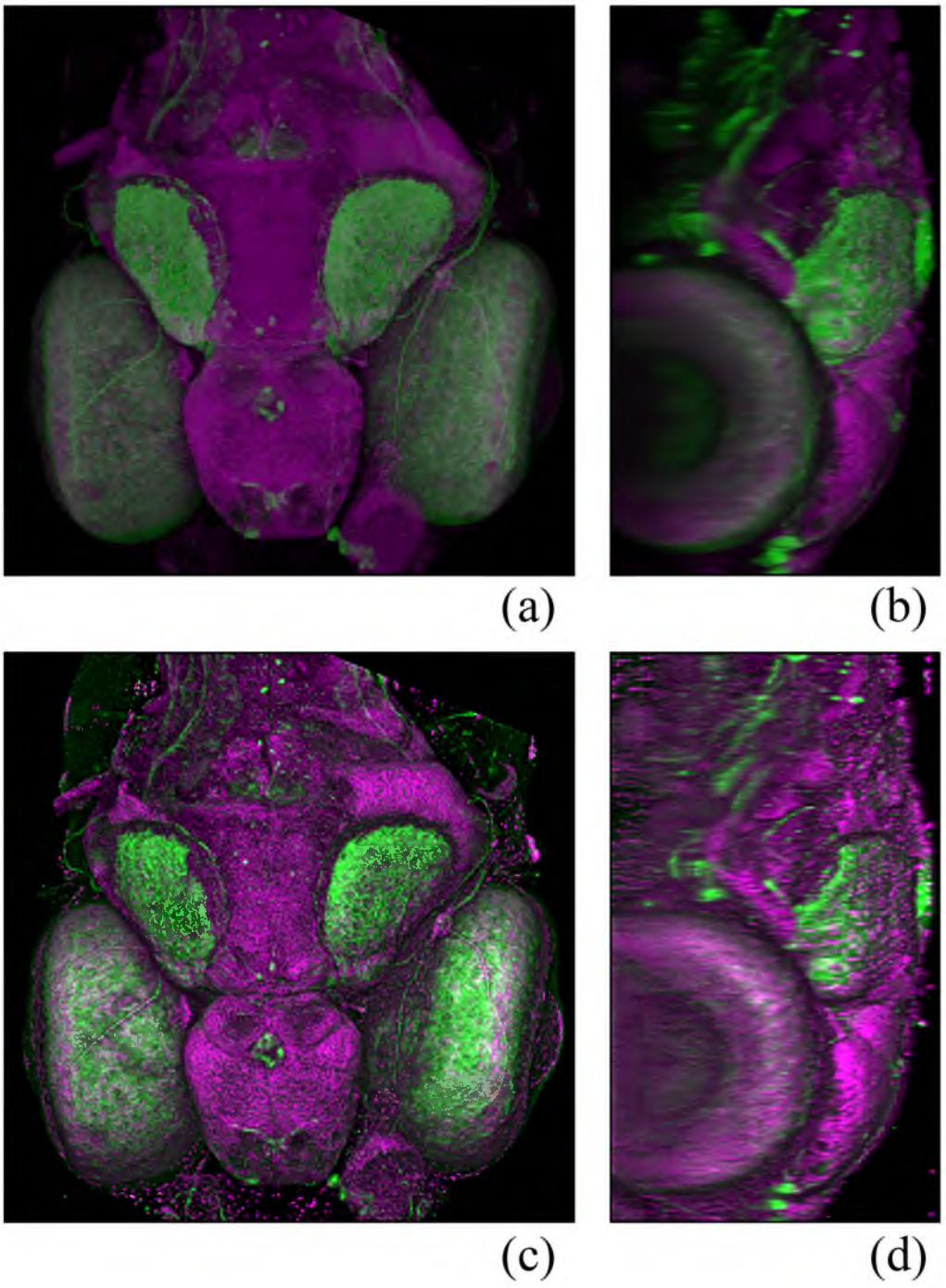

Figure 3.12. Results of scale-space equalization. (a) Dorsal view of a zebrafish head dataset rendered without any enhancement. (b) Lateral view of a zebrafish head dataset rendered without any enhancement. (c) Dorsal view of a zebrafish head dataset rendered with enhancements applied. (d) Lateral view of a zebrafish head dataset rendered with enhancements applied.

low intensity signals. Color mapping is an effective and intuitive method, but not without problems for normal volume renderings. Figure 3.13(a) shows a confocal channel rendered with a rainbow colormap as the transfer function. Biologist users often feel that it does not fit into their research purposes well, because the colors in the result do not clearly correspond to those in the colormap and voxels with high scalar intensities, which represent strong biological expression and are important to the research, are mostly occluded.

The 2D color mapped maximum intensity projection solves both problems stated above. Figure 3.13(b) shows the result of a 2D color mapped MIP with the same colormap as in Figure 3.13(a). Since MIP does not use normal volume compositing, the colors of its result represent the exact intensity values of the voxels, and high intensity voxels are always visualized. In fact, MIP is the only method recognized by biologists for inspecting fluorescent staining intensities.

However, the way that MIP renders volume data can cause two problems for users. First, the orientation of a volume dataset under examination becomes obscure, which may confuse users especially when they rotate the data. Biologists usually prefer orthographic over perspective projection in order to better compare

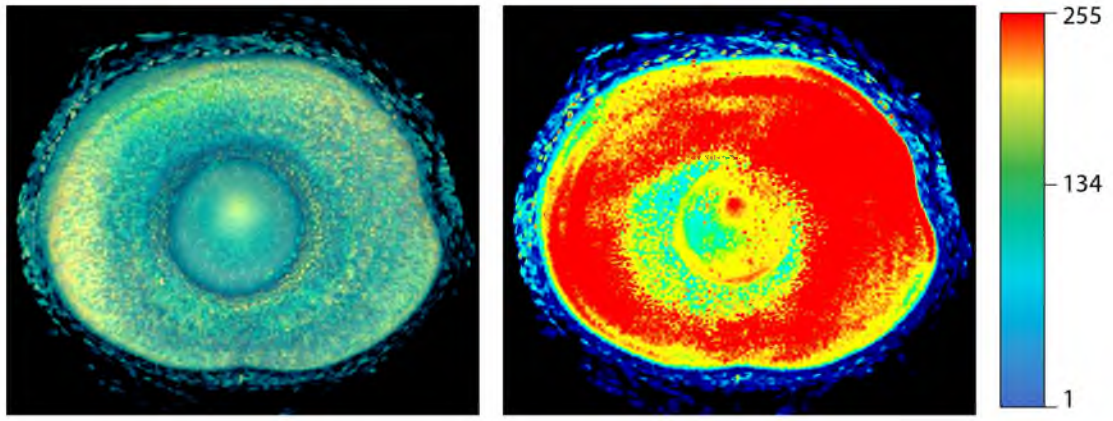

Figure 3.13. Using a colormap as the volume transfer function and 2D color mapping of the MIP. All results have the same colormap, as shown on the right. (a) The colormap is used as the volume transfer function. (b) The colormap is applied to the MIP rendering output. The dataset shows a 5 dpf zebrafish eye.

structure sizes. It worsens the problem of orientation perception when MIP is used. Second, details of surface structures are lost, because unlike most other volumetric data, the structural details of confocal data are always comprised of less intensive signals surrounding high-intensity ones, since the signals are generated by fluorescence emission. Adding global lighting effects, such as shadows, can help orient viewers to the renderings of volume data, thus solving the first problem. The second problem can be solved by incorporating local lighting effects, such as Phong shading. There are methods such as two-level volume rendering [38] and MIDA (Maximum Intensity Difference Accumulation) [12] that combine the advantages of MIP and shading effects from direct volume rendering. However, the results of above techniques are both somewhere between MIP and DVR. One important feature of MIP that biologists appreciate, especially when a colormap is used, cannot be ensured, i.e., colors of final result represent the exact intensity values of the voxels. Furthermore, how global lighting effects, such as shadows, can be applied with above techniques is not clear. Fortunately, one structure in confocal data is always comprised of low intensity details surrounding high intensity cores. This simplification of structures enables us to render MIP and lighting effects separately, and then combine them with 2D compositing. The MIP pass is color-mapped for examining the biological expression amount of structure cores; the shading and shadow passes render surface details and enhance orientation perception. The 2D compositing is completed by modulating the color brightness of the MIP rendering with the brightness of the effect passes.

Figure 3.14 illustrates the 2D compositing and its result of a confocal dataset rendered with a shading layer. In addition, a shadow layer can be rendered and composited similarly. Biologists can use the 2D color-mapped MIP with overlays for inspecting the amount of biological expression, because the result has a correct color correspondence with the colormap used. The renderings of shading and shadow passes are grayscale images, and only the color brightness of the 2D color-mapped MIP is modulated; therefore, the color hue stays the same, which is the actual variable used in the colormap. Figure 3.15 compares 2D compositing

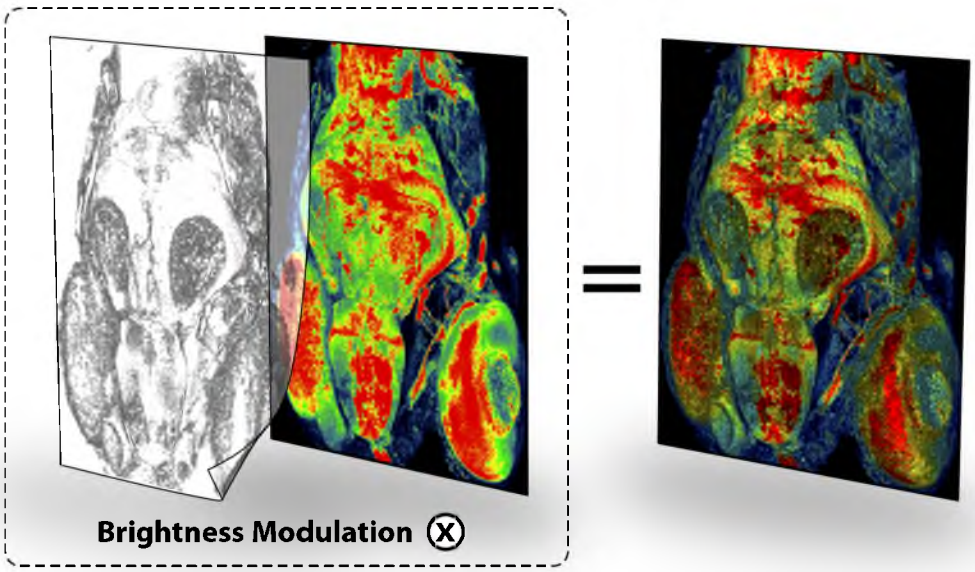

Figure 3.14. A shading pass is composited with the result of a 2D color-mapped MIP pass. The result has the advantages of both MIP and DVR. The dataset has three confocal channels, including stained muscles, neurons, and nuclei.

with DVR and MIDA [12], which uses a modified volume compositing scheme. Since other methods use compositions in 3D, the voxel colors are blended and cannot match the colors used in the colormap. However, for complex structures such as a network of blood vessels, this method has its limitation: shading/shadow and MIP cannot always be rendered consistently, since users have to adjust the volume transfer function for shading and shadow layers. In practice, this mode is used when important features are best represented by MIP and biologist users want to add enhancements for surface details and orientation perception.

3.7 The FluoRender Visualization Pipeline

We use OpenGL and GLSL for the implementations of the techniques discussed previously, including on the fly evaluation of the volume transfer function, tone-mapping operator evaluations, shading and shadow calculations, compositing, and color mapping. While most of the implementations should be straightforward, there are some details worth mentioning. The three tone-mapping operators can concatenate and be evaluated at once - we first generate the scale space, apply

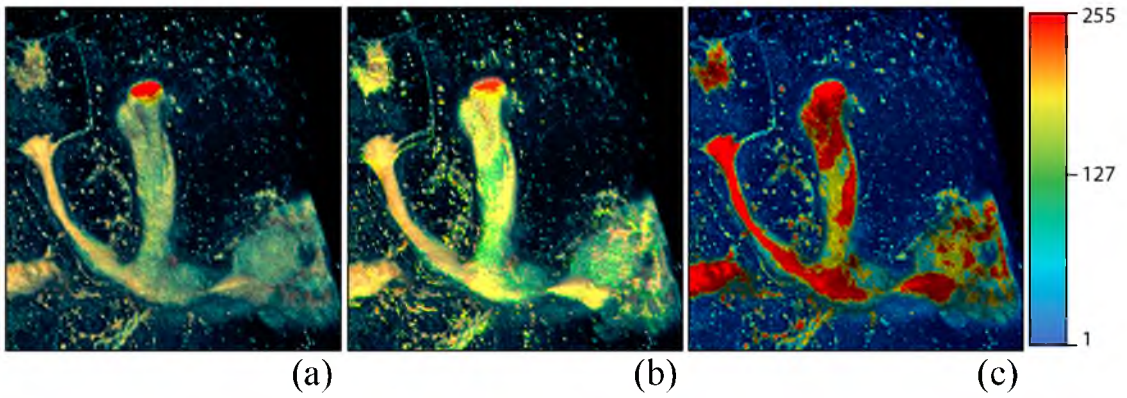

Figure 3.15. A comparison of DVR (a), MIDA (b), and shading overlay on MIP (c). They all use the same colormap shown on the right. The dataset is the mushroom body (MB) of an adult *Drosophila*, stained with nsyb::GFP. This fluorescent protein specifically binds to presynaptic regions of neurons. Thus, higher signal intensity indicates higher density of synapses of the mushroom body. By using MIP with 2D overlays, we can clearly see the head of α/α' lobe has higher presynaptic density than its neck, which can be similarly observed for β/β' lobe.

gamma and luminance adjustment to all the levels, and then calculate equalization. For fast processing speed, we use the built-in mipmap generating function of OpenGL to approximate the scale space. For shadow overlay calculation, we use a 2D image space method similar to that of depth buffer unsharp masking [65]. Unlike other confocal visualization tools, such as Imaris and Volocity, which use ray tracing to precalculate shadows and are not real-time, we use 2D filtering on the depth buffer. The rendering speed is real-time, which helps when multichannel and time-sequence datasets are visualized. The 2D image space methods are easily modularized, and each module can work independently of another. However, building an integrated visualization system that neurobiologists can easily use, especially when the amount of datasets visualized is large, still requires meticulous design of its user interactions. We developed the user interactions through close cooperation with frequent FluoRender users and experts in confocal microscopy. Figure 3.4 shows a screen capture of FluoRender's main user interface.

Figure 3.16 is an illustration of the FluoRender visualization pipeline. The leftmost blocks represent data inputs. Multiple confocal channels can be loaded

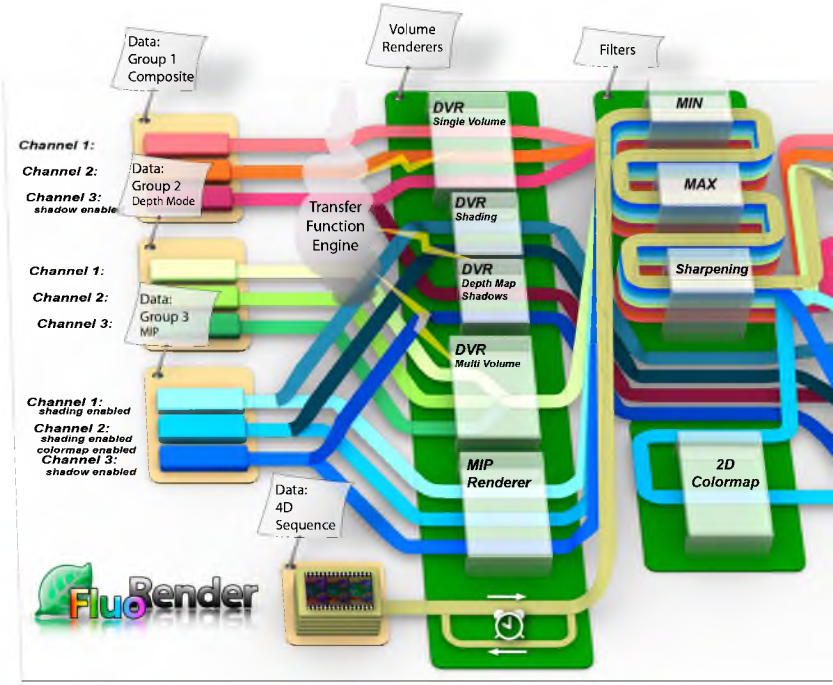

Figure 3.16. The visualization pipeline of FluoRender.

Tone Mappers
Compositor
Overlays
GAMMA
LUMINANCE
EQUALIZATION
Blending
Blending
Blending
Output

and visualized at the same time. They are grouped by users for easy organization and adjustment. Time-sequence datasets can be loaded as well. The confocal channels are first processed by different types of volume renderers, which generate internal renderings by direct volume rendering or maximum intensity projection. Channels in depth mode are combined and rendered with a separate volume renderer. Effect layers such as shading and shadow are also generated within this process. According to settings of each channel, a rendering result then goes through the modules of 2D image space enhancements, including filtering, color mapping, overlay compositing, and tone mapping. The enhanced results are combined according to their group/view settings, such as layered or composite modes. The final result is output to the viewport of the FluoRender main user interface. For time-sequence data, they are processed with the same pipeline. A timepoint of a sequence is read and fed into the pipeline each time according to an event-driven mechanism.

CHAPTER 4

SEGMENTATION AND ANALYSIS OF CONFOCAL MICROSCOPY DATA

4.1 Introduction to Morphological Diffusion

For interactive speed of confocal volume segmentation, we propose morphological diffusion on a mask volume for selecting desired structures. Morphological diffusion can be derived as one type of anisotropic diffusion under the assumption that energy can be nonconserving during transmission. Its derivation uses the results from both anisotropic diffusion and mathematical morphology.

4.1.1 Diffusion Equation and Anisotropic Diffusion

The diffusion equation describes energy or mass distribution in a physical process exhibiting diffusive behavior. For example, the distribution of heat (u) in a given isotropic region over time (t) is described by the heat equation:

$$\frac{\partial u(\mathbf{x},t)}{\partial t} = c\nabla^2 u(\mathbf{x},t) = \nabla \cdot (c\nabla u(\mathbf{x},t)) \tag{4.1}$$

In Equation 4.1, c is a constant factor describing how fast temperature can change within the region. We want to establish a relationship between heat diffusion and morphological dilation. First, we look at the conditions for a heat diffusion process to reach its equilibrium state. Equation 4.1 simply tells us that the change of temperature equals the divergence of the temperature gradient field, modulated by a factor c. We can then classify the conditions for the equilibrium state into two cases:

- Zero gradient. Temperatures are the same everywhere in the region.

• Solenoidal (divergence-free) gradient. The temperature gradient is nonzero, but satisfies the divergence theorem for an incompressible field, i.e., for any closed surface within the region, the total heat transfer (net heat flux) through the surface must be zero.

The nonzero gradient field can be sustained because of the law of conservation of energy. Consider the simple 1D case in Figure 4.1, where the temperature is linearly increasing over the horizontal axis. For any given point, it gives heat out to its left neighbor with lower temperature and simultaneously receives heat of the same amount from its right neighbor. In this 1D case, the loss and gain of heat reach a balance when the temperature field is linear. As we are going to see later, if we lift the restriction of energy conservation, the condition for equilibrium may not hold, and we need to rewrite the heat equation under new propositions.

The generalized diffusion equation is anisotropic. Specifically, we are interested in the anisotropic diffusion equation proposed by Perona and Malik [80], which has been extensively studied in image processing.

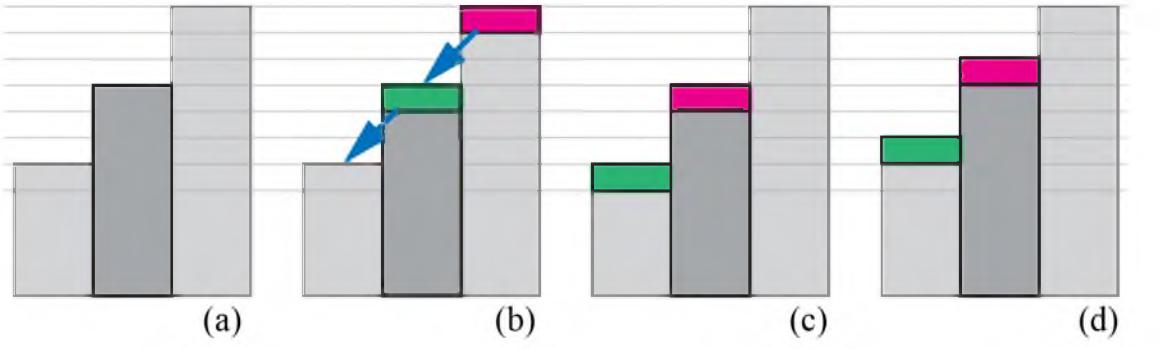

Figure 4.1. Conserving and nonconserving energy transmissions. (a) The initial state has a linear gradient. We are interested in the energy change of the center piece. (b) Energy is transferred from high to low (gradient direction), as indicated by the arrows. (c) Result of typical conserving transmission. The center piece receives and gives the same amount of energy, which maintains a solenoidal gradient field. (d) Result of dilation-like transmission, which is not energy conserving. The center piece gains energy and a solenoidal gradient field cannot be sustained.

$$\frac{\partial u(\mathbf{x},t)}{\partial t} = \nabla \cdot (g(\mathbf{x},t)\nabla u(\mathbf{x},t)) \tag{4.2}$$

In Equation 4.2, the constant c in the heat equation is replaced by a function $g()$, which is commonly calculated in order to stop diffusion at high gradient magnitude of u.

4.1.2 Morphological Operators and Morphological Gradients

In mathematical morphology, erosion and dilation are the fundamental morphological operators. The erosion of an image I by a structuring element B is:

$$\varepsilon(x) = min(I(x+b)|b \in B) \tag{4.3}$$

And the dilation of an image I by a structuring element B is:

$$\delta(x) = max(I(x+b)|b \in B) \tag{4.4}$$

For a flat structuring element B, they are equivalent to filtering the image with minimum and maximum filters (rank filters of rank 1 and N, where N is the total number of pixels in B), respectively.

In differential morphology, erosion and dilation are used to define morphological gradients, including Beucher gradient, internal, and external gradients, etc. Detailed discussions can be found in [84] and [95]. In this chapter, we are interested in the external gradient with a flat structuring element, since for confocal data, we always want to extract structures with high scalar values and the region-growing process of high scalar values resembles dilation. Thus, the morphological gradient used here is:

$$|\nabla I(x)| = \delta(x) - I(x) \tag{4.5}$$

Please note that for a multivariable function I, Equation 4.5 is essentially a discretization scheme for calculating the gradient magnitude of I at position $\mathbf{x}$.

4.1.3 Morphological Diffusion

If we consider the morphological dilation defined in Equation 4.4 as energy transmission, it is interesting to notice that energy is not conserved. In Figure 4.1, we show that within a neighborhood of a given position, the local maximum can give out energy without losing its own. Thus, for a closed surface within the whole region, the net energy flux can be non-negative. In other words, under the above assumption of nonconserving energy transmission, the solenoidal gradient condition (Section 3.1) for the equilibrium of heat diffusion no longer holds. Therefore, the heat diffusion can only reach its equilibrium when the energy field has zero gradients.

Based on the above reasoning, we can rewrite the heat equation (Equation 4.1) to its form under the dilation-like energy transmission:

$$\frac{\partial u(\mathbf{x},t)}{\partial t} = c|\nabla u(\mathbf{x},t)| \tag{4.6}$$

Equation 4.6 can be simply derived from Fourier's law of heat conduction [16], which states that heat flux is proportional to negative temperature gradient. However, we feel our derivation can better reveal the relationship between heat diffusion and morphological dilation. To solve this equation, we use forward Euler through time and the morphological gradient in Equation 4.5. Notice that the time step Δt can be specified with c for simplicity when the discretization of time is uniform. Then, the discretization of Equation 4.6 becomes:

$$\begin{aligned} u_{i+1}(x) &= u_i(x) + c(\delta_i(x) - u_i(x)) \\ &= c\delta_i(x) + (1-c)u_i(x) \end{aligned} \tag{4.7}$$

When $c = 1$, the trivial solution of Equation 4.6 becomes the successive dilation of the initial heat field, which is exactly what we expected.

Thus, we have established the relationship between morphological dilation and heat diffusion from the perspective of energy transmission. We name Equation 4.7 morphological diffusion, which can be seen as one type of heat diffusion process

under nonconserving energy transmission. Though a similar term has been used in the work of Segall and Acton [92], we use morphological operators for the actual diffusion process rather than calculating the stopping function of anisotropic diffusion. Our purpose of using the result for interactive volume segmentation rather than simulating physical processes legitimizes the lifting of the requirement for conservation. We are interested in the anisotropic version of Equation 4.7, which is obtained simply by replacing the constant c with a stopping function $g(x)$:

$$\begin{aligned} u_{i+1}(x) &= u_i(x) + g(x)(\delta_i(x) - u_i(x)) \\ &= g(x)\delta_i(x) + (1 - g(x))u_i(x) \end{aligned} \tag{4.8}$$

In Equation 4.8, when the stopping function $g(x)$ is in $[0,1]$, the iterative results are bounded and monotonically increasing, which lead to a stable solution. By using morphological dilation (i.e., maximum filtering), morphological diffusion has several advantages when applied to confocal data and implemented with graphics hardware. Morphological dilation's kernel is composed of only comparisons and has the least computational overhead. The diffusion process only evaluates at nonlocal maxima, which are forced to reach their stable states with fewer iterations. Last but not least, in an iterative process of morphological diffusion evaluation, since local scalar values are increasing monotonically, the converging result is stable when neighboring voxels are updated simultaneously by multiple threads. This means this algorithm is very suitable to be implemented on massive parallel graphics hardware. No extra memory and context switch are necessary for the process known as frame buffer object ping-pong, which is commonly used in evaluation of standard anisotropic diffusion on graphics hardware. In Chapter 5, we further discuss frame buffer feedback loops, which is considered efficient for evaluating converging iterative processes. When coupled with our user interactions, morphological diffusion is able to extract desired structures from typical confocal data with interactive speed on common PCs.

4.2 User Interactions for Interactive Volume Segmentation

Paint selection [74], [61] with brush strokes is considered one of the most useful methods for 2D digital content authoring and editing. Incorporated with segmentation techniques, such as level set and anisotropic diffusion, it becomes more powerful yet still intuitive to use. For most volumetric data, this method becomes difficult to use directly on the renderings, due to occlusion and the complexity of determining the depth of the selection strokes. Therefore, many volume segmentation tools' user interactions are limited to 2D slices. Taking advantage that the confocal channels usually have sparsely distributed structures, direct paint selection on the render viewport is actually very feasible, though selection mistakes caused by occlusion cannot be completely avoided. Using the result from Section 4.1, we developed interaction techniques that let users progressively select structures from confocal data. These techniques share similarities with the sketch-based volume selection methods described in previous literature [113], [18], [78], [1]. However, the algorithm presented in Section 3 allows us to use paint strokes with varying sizes so that users can progressively select structures and edit the selections.

Figure 4.2 illustrates the basic process of extracting a neural structure from confocal volume with our method. First, a scalar mask volume is generated. Then, the user defines seed regions by painting on the render viewport. The pixels of the defined region are then projected into 3D as a set of cones (cylinders if the viewport is orthographic) from the camera's viewpoint. Voxels within the union of these cones are thresholded to generate seeds in the mask volume, where seeds have the maximum scalar value, and other voxels have zero value. Then a wider region, which delimits the extent of subsequent diffusion, is defined by painting again on the viewport. The second region is projected into the volume similarly. Then, in the mask volume, the selected seeds propagate by iteratively evaluating Equation 4.8. Structures connected to those registered by the seeds are then selected in the mask volume. The resulting mask volume is not binary,

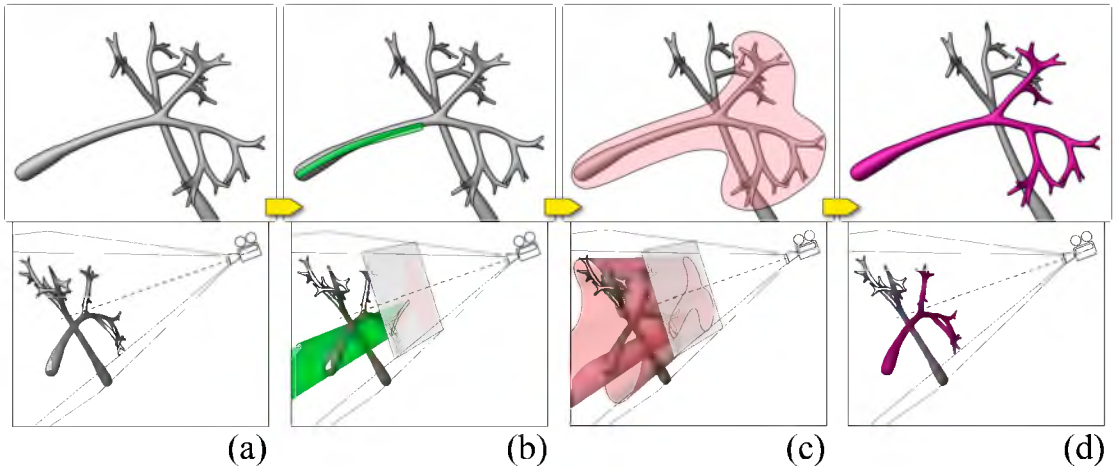

Figure 4.2. Volume paint selection of neural structures from a confocal volume. (a) The visualization of certain neural structures (top) and the camera setup (bottom). (b) A user paints on the viewport. The stroke (green) is projected back into the volume to define the seed generation region. (c) The user paints on the viewport to define the diffusion region. The stroke (red) is projected similarly and the seeds generated in B grow to either the structural boundaries or the boundary defined by the red stroke. (d) The intended neural structure is extracted.

though the structural boundaries can be more definitive by adjusting the stopping function, which is subsequently discussed. After each stroke, the mask volume is instantly applied to the original volume, and the selected structures are visualized with a different color against the original data. The user can repeat this process for complex structures, since the calculation only modifies the mask volume and leaves the original data intact.

We use gradient magnitude ($|\nabla V|$) as well as scalar value (V) of the original volume to calculate the stopping function in Equation 4.8, since, for confocal data, important structures are stained by fluorescent dyes, and they should have high scalar values. The stopping function (Equation 4.9) is the product of two parts. $g_1()$ is calculated as the Gaussian of $|\nabla V|$, which stops the growth at high gradient magnitude values; $g_2()$ is calculated as the Gaussian of V, which stops the growth at low scalar intensities. The combined effect of the two parts is that the growth stops at regions of both high gradient magnitude values and low intensities, which are considered edges or boundaries for confocal data.

$$
\begin{aligned}
g(V) &= g_1(V) \cdot g_2(V) \\
g_1(V) &= \begin{cases} 1 & |\nabla V| < t_1 \\ e^{-\frac{(|\nabla V| - t_1)^2}{k_1^2}} & otherwise \end{cases} \\
g_2(V) &= \begin{cases} e^{-\frac{(V - t_2)^2}{k_2^2}} & V < t_2 \\ 1 & otherwise \end{cases}
\end{aligned}
\tag{4.9}
$$

The graphs of the two parts of the stopping function are in Figure 4.3. t_1 and t_2 translate the falloffs of $g_1()$ and $g_2()$, and the falloff steepness is controlled by k_1 and k_2. The combined effect of $g_1()$ and $g_2()$ is that the seed growing stops at high gradient magnitude values and low intensities, which are considered borders for structures in confocal data.

By limiting the seed growth region with brush strokes, users have the flexibility of selecting the desired structure from the most convenient angle of view. Furthermore, it also limits the region for diffusion calculations and ensures real-time interactions. For less complex structures, seed generation and growth region definition can be combined into one brush stroke; for over-segmented or mistakenly selected structures, an eraser can subtract the unwanted parts. We

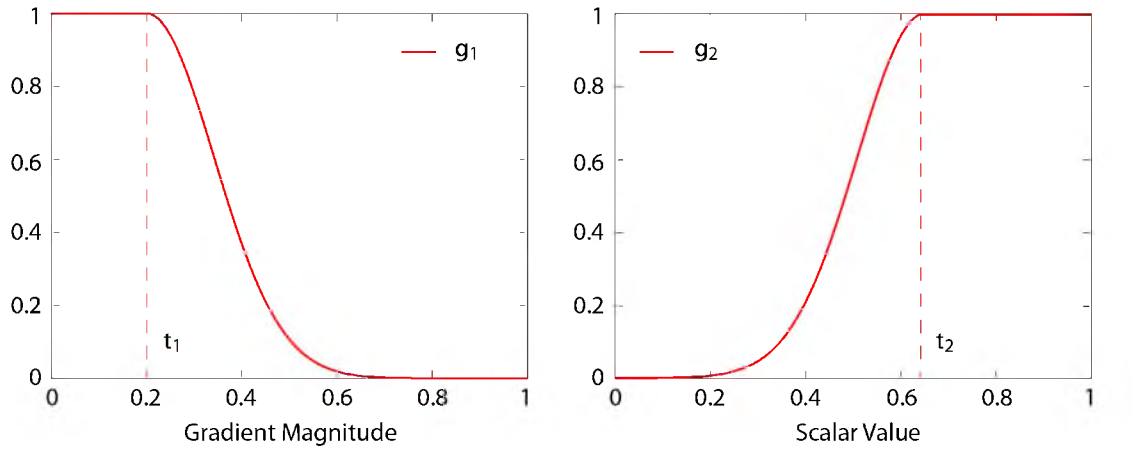

Figure 4.3. The two parts of the stopping function. $g_1()$ is for stopping the growth at high gradient magnitude values and $g_2()$ is for stopping at low scalar intensities. The final stopping function is the product of $g_1()$ and $g_2()$.

designed three brush types for both simplicity and flexibility. Biologists can use these brushes to extract different structures from confocal data.

Selection brush combines the definition of seed and diffusion regions in one operation. As shown in Figure 4.4, it has two concentric circles in the brush stamp shape. Strokes created by the inside circle are used for seed generation, and those created by the outside circle are for diffusion region definition. Usually, the diameter of the inside circle is set slightly smaller than the root of a structure. The diameter of the outside circle is determined by how the substructures branch out from the root structure. By combining the two operations, it makes interaction easier. For example, to extract an axon and its terminal branches, the inside circle is set roughly to the size of the axon, and the outside circle is set to that can enclose the terminals. Morphological diffusion is calculated on finishing each stroke, which appends newly selected structures to existing selections. Since users can easily rotate the view while painting, it is helpful to use this tool and select multiple structures or different parts of one complex structure from the most convenient observing directions. Figure 4.4 demonstrates using the selection brush to extract a visual projection neuron of a *Drosophila* brain.

Eraser behaves similarly to the selection brush, except that it first uses morphological diffusion to select structures, and then subtracts the selection from previous results. The eraser is an intuitive solution to issues caused by occluding structures: mistakenly selected structures because of obstruction in 2D renderings can usually be erased from a different angle of view. Figure 4.5 demonstrates such a situation where one neuron obstructs another in the rendering result. The eraser is used to remove the mistakenly selected structures.

Diffusion brush only defines the diffusion region. It generates no new seeds and only diffuses existing selections within the region defined by its strokes. Thus, it has to be used after the selection brush. With the combination of the selection brush and the diffusion brush, occluded or occluding neural structures can be extracted easily, even without changing viewing angles. Figure 4.6 shows the same example as in Figure 4.5. First, the selection brush is used to extract only the

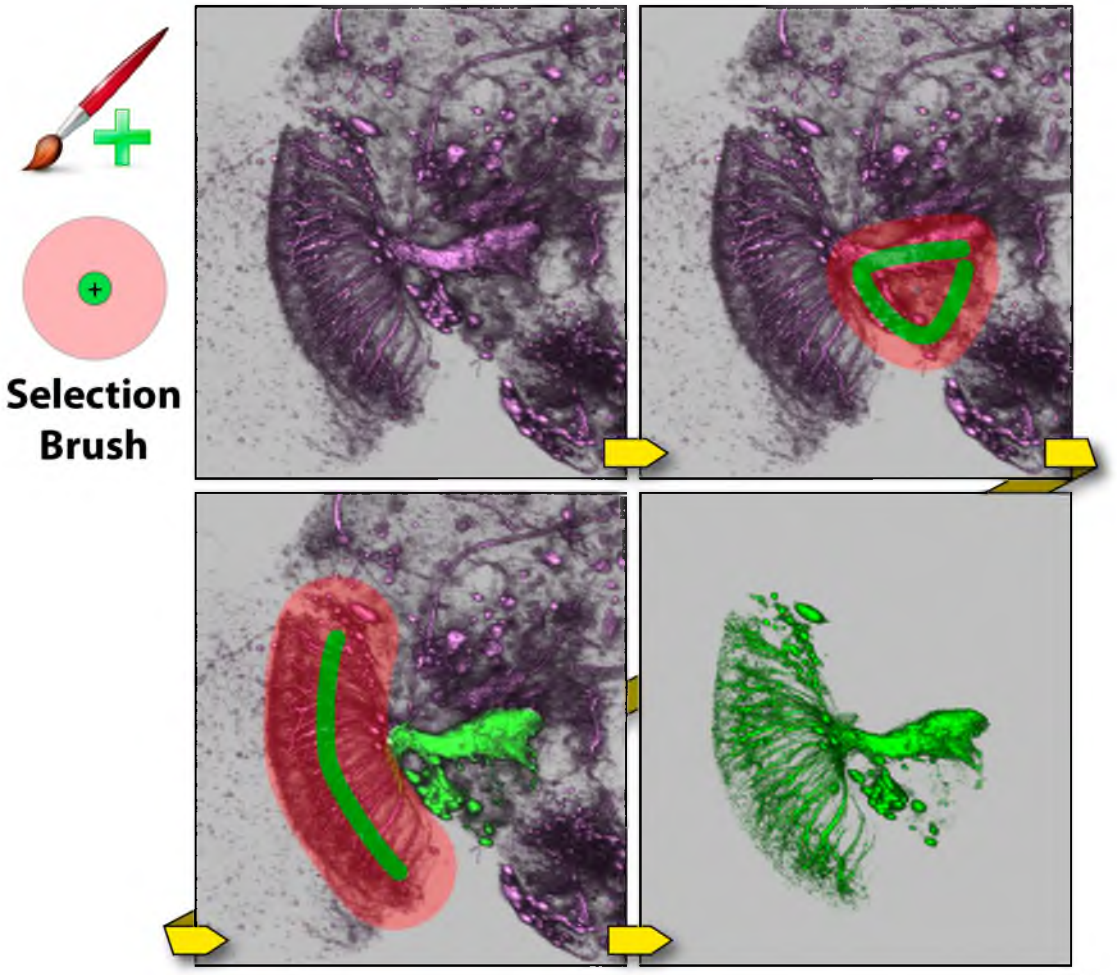

Figure 4.4. Selection brush. The dataset contains neurons of a *Drosophila* adult brain. The original dataset has a neuron that a user wants to extract, which is visual projection neuron LC14 [76]. First, a stroke is painted with the selection brush. Then a second stroke is painted, which covers the remaining part of the neuron. Finally the neuron is extracted.

nonobstructing part of the neuron. Then, the remaining of the neuron is appended to the selection by painting with the diffusion brush. Since the obstructing part is not connected to the neuron behind, and the diffusion brush does not generate new seeds in that region, the neuron behind is not selected.

As seen in the above examples, our interactive segmentation scheme allows inaccurate user inputs within fairly good tolerance. However, using a mouse to conduct painting work is not only imprecise but also causes fatigue. We support

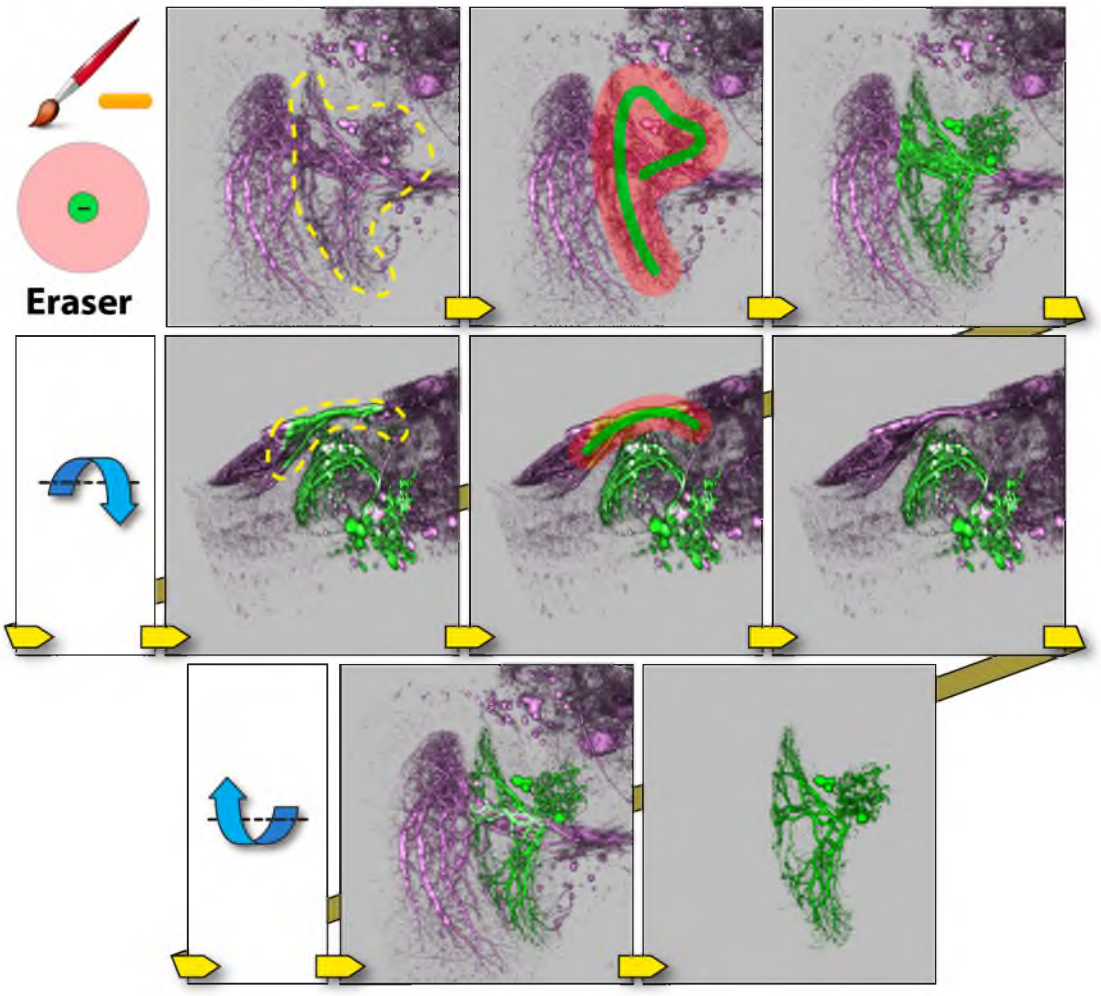

Figure 4.5. Eraser. The dataset contains neurons of a *Drosophila* adult brain. The yellow dotted region indicates the structure that a user wants to extract (visual projection neuron LT1 [76]). From the observing direction, the structure obstructs another neuron behind (visual projection neuron VS [76]). First, a stroke is painted with the selection brush. Then, LT1 is extracted, but VS is partially selected. Then, the view is rotated around the lateral axis. The second yellow dotted region indicates extra structures to be removed. Another stroke is painted with the eraser. The extra structures are then removed. The view is rotated back. Finally, we have a visualization of the extracted neuron (LT1).

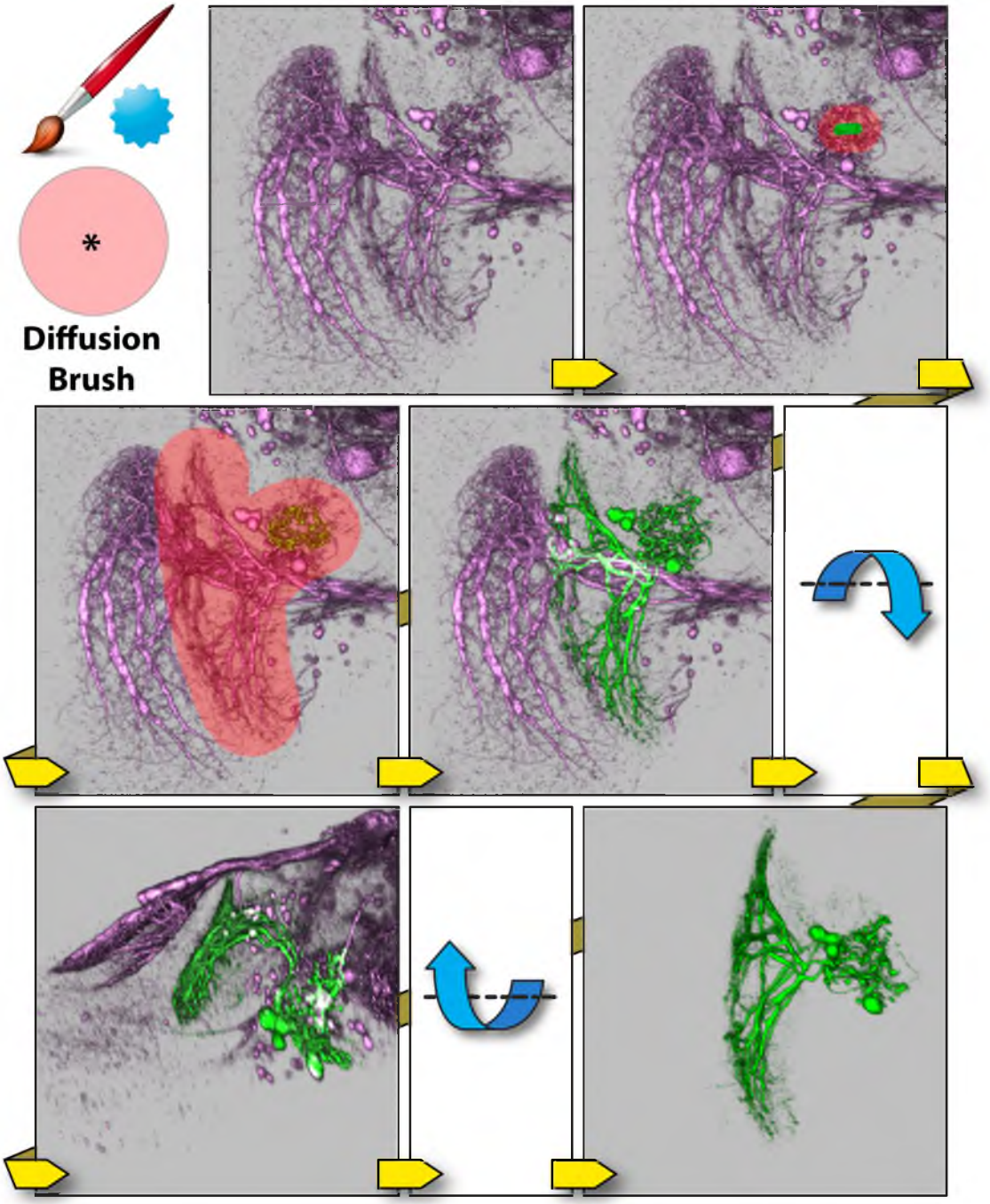

Figure 4.6. Diffusion brush. The dataset contains neurons of a *Drosophila* adult brain. The original dataset is the same as in Figure 4.5. First, a stroke is painted with the selection brush on the nonobstructing part of LT1 and part of LT1 is selected. Then, the diffusion brush is used to select the remaining of LT1. LT1 is selected without selecting the obstructed neuron (visual projection neuron VS). Then, the view is rotated around the lateral axis, to confirm the result. Finally, we have a visualization of neuron LT1 after extraction.

the latest digital tablets in our tool for dexterity enhancement. The active tablet area is automatically mapped to the render viewport. Thus, all the available area on the tablet is used in order to maximize the precision, and the user can better estimate the location of the strokes even when the stylus is hovering above the active area of the tablet. Furthermore, stylus pressure is utilized to control the brush size. Though the pressure sensitive brushes are a feature that can be turned off by users, our collaborating neurobiologists like the flexibility of changing the brush sizes on the fly. It helps to extract neural structures of varying sizes more precisely (Figure 4.7).

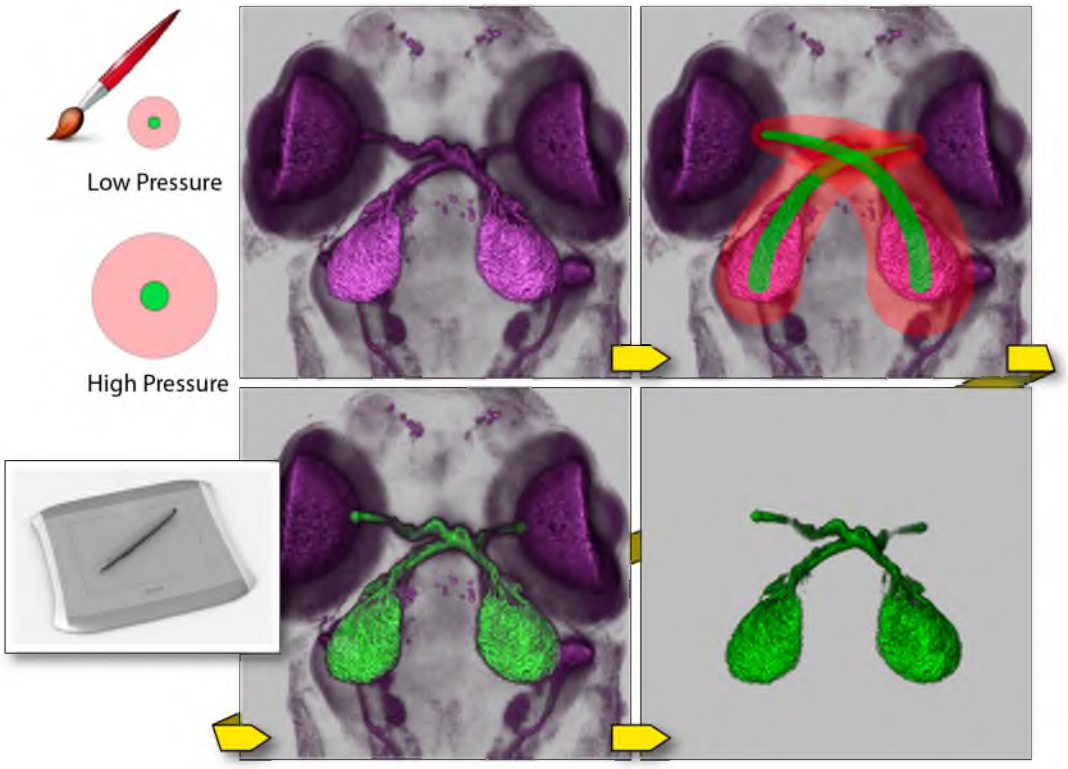

Figure 4.7. A digital tablet and its usage. The dataset contains neurons of a zebrafish head. The original dataset contains stained tectum lobes and photoreceptors of eyes. Since the tectum lobes and the photoreceptors actually connect, we want to better control the brush size for diffusion at the regions of connection, when only the tectum lobes are to be extracted. First, two strokes are painted with the selection brush. The stroke size changes as user varies the pressure applied to the tablet's stylus. The tectum lobes are then selected. Finally, the tectum lobes are extracted and visualized.

4.3 Integration of Interactive Segmentation with Visualization Functions

In FluoRender, the previously discussed interactive segmentation techniques are not simply placed on top of its existing visualization pipeline. Instead, they take advantage of the features within the visualization pipeline and give users more intuitive operations. On one hand, a clear visualization improves segmentation accuracy. On the other hand, segmented results are used to enhance visualization. In an integrated workflow of FluoRender, the interactions between visualization and segmentation include the following techniques.

• Volume transfer function. This is only obvious in an interactive environment. When users adjust parameters of the volume transfer function for a clear visualization, sometimes it may only work for a simple structure or one part of an entire dataset. Segmentation can take the values emphasized or suppressed by the transfer function instead of the original values. Structures emphasized by current transfer function settings are more easily extracted. Then, for a different structure or different part of the same dataset, a different transfer function can be set before segmentation. The results should satisfy users' intentions better and segmentation functions seem to be more intuitive and versatile for users. Figure 4.8 shows an example of an extreme case, which requires us to extract both bones and muscles from a confocal dataset. The muscles, tendons, and nerves of this specimen were stained; bones could be visualized as black regions. While it is relatively easy to extract the muscles, correct segmentation of bones becomes difficult, as their boundaries could not even be clearly visualized without transfer function manipulations. Here, we first inverted the scalar values of the dataset, and then decreased the gamma in the volume transfer function. Not only were the bones rendered with brighter intensities, their boundaries were also easily enhanced with 2D image space methods. After some simple adjustments, these structures were easily identified and extracted using FluoRender.

• Tone-mapping operators. In Section 3.5, Chapter 3, we discussed tone-mapping operators in FluoRender. Their effects on volume rendering results can

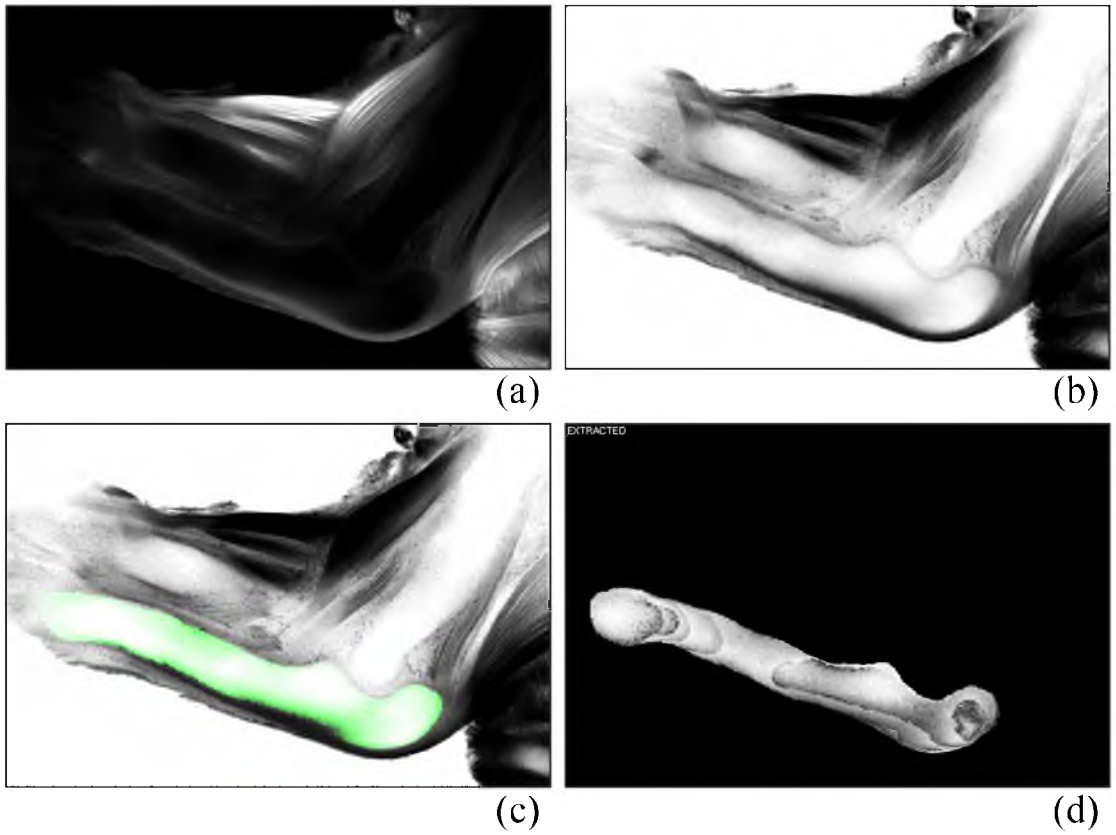

Figure 4.8. Segmenting the ulna from the muscle channel of a confocal scan of a mouse embryo. (a) Bones are black regions in the origin channel. (b) The scalar intensities of this channel are inverted. (c) We segment the ulna by painting. (d) the segmented result.

be applied to segmentation results as well. When users change the tone-mapping settings in FluoRender, we first generate a weight map, which contains the brightness difference by dividing the original rendering from the tone-mapped result. The weight map is then projected along with painted strokes. When the seeds are generated or the morphological diffusion is evaluated, the volume transfer function adjusted voxel values are again modulated by the weights on the weight map. Therefore, structures with intensity voxels are enhanced and can be extracted more easily without changing the settings for morphological diffusion. Figure 4.9 is an illustration of this process.

- Clipping planes. FluoRender has a specially designed clipping system to

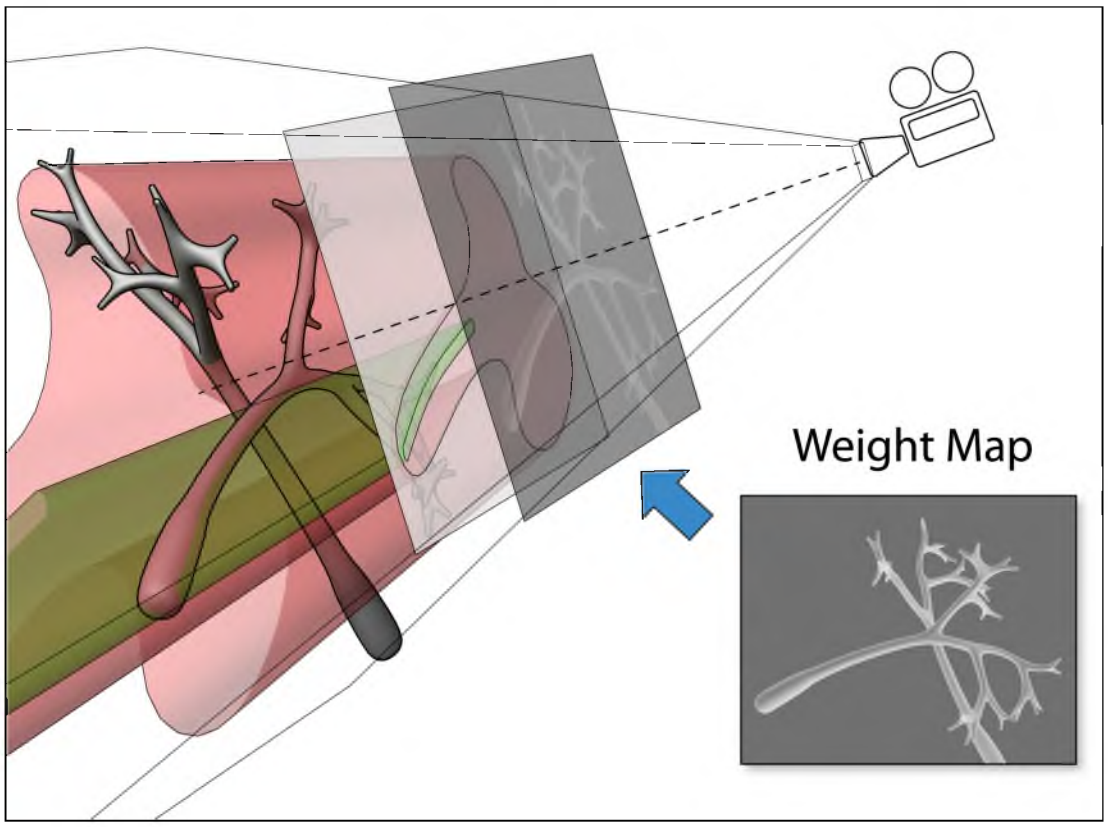

Figure 4.9. A weight map is the brightness difference between original rendering and tone-mapped result. It is projected into the volume along with paint strokes. Both seed generation and diffusion are then influenced by the weight map.

work along with its interactive segmentation. Firstly, it uses six axis-aligned clipping planes, whose positions along the axes can be adjusted individually. An arbitrary block of volume data can be extracted using these clipping planes. Then, segmentation is only calculated within this isolated region. Secondly, the six-plane system allows quickly setting two opposite clipping planes to include just one section of the original volume data. This is equivalent to the commonly used slice-based segmentation, when rotation of the dataset is then locked to only orthogonal views. Thirdly, FluoRender allows users to rotate the clipping planes. It becomes a better integrated implementation of the oblique slicing plane from Sowell et al. [96], since the oblique image plane is embedded into the functions of clipping planes. Finally, the clipping plane system allows aligning with current

viewing plane. Using view direction to rotate clipping planes has the advantage that users do not need to interact with less intuitive widgets for rotations. They can simply change the current viewing direction and then tell the clipping planes to align with the best view.

• Multichannel calculations. There are often correlated structures in different channels from one confocal scan. For example, cell membranes confine nuclei into clearly separate regions. Such spatial relationships not only help visual identification and distinction structures, but can also be incorporated into segmentation calculations. We define four calculations between two correlated confocal channels to utilize their spatial relationship when possible.

◦ *Addition.* It adds corresponding voxel scalar values of two channels. The result has enhanced common structures. Or it is simply used to combine two channels.

◦ *Subtraction.* It subtracts corresponding voxel scalar value of one channel from another. It is used to remove structures common to two channels that are unwanted in one channel. In the cell membrane and nucleus example, the membrane channel is subtracted from the nucleus channel for better definition of each nucleus.

◦ *Division.* It divides corresponding voxel scalar value of one channel by another. It is often used to compare the difference between two channels.

◦ *Boolean AND.* For scalar data, this operation is actually calculating the minimum of two corresponding voxels. It is used to extract common structures from two confocal channels. These structures are usually stained by multiple fluorescent tags and termed colocalization in biology. It is referred as Boolean AND because of the similar effect to that of binary data.

In the user interface of FluoRender, these operations are placed together with segmentation settings. Users can perform them before segmenting confocal channels or after for comparison.

CHAPTER 5

CELL TRACKING USING SYNTHETIC BRAINBOWS

Interactive techniques presented in Chapter 4 were designed to work best with 3D branching structures such as neuron cells and nerve bundles. Most cells in biology research have simpler shapes but occur more frequently. Furthermore, continuous confocal imaging of living cells becomes available. Biologists often want to find the trajectories of a group of cells through time. Our techniques designed for extracting neural structures need to be adapted and improved for cell identification and tracking. We developed a technique, Synthetic Brainbows, assigning random IDs to complex structures in confocal data. Its results resemble those from the true Brainbow technique. We first tested and determined randomness in GPU framebuffer feedback loops. Then, we applied GPU framebuffer feedback loops to single-channel confocal data to generate labeled volumes. Finally, the method was applied to time sequences of confocal data to track moving cells.

5.1 Randomness in a GPU Framebuffer Feedback Loop

As discussed in Section 2.5, we expect randomness in asynchronous operations in the massive parallel copmuting environment of GPUs. In order to leverage the randomness, we would like to first examine the behavior of GPU framebuffer feedback loops. To avoid unnecessary complexity, we restrict the investigation to integer textures and turn off texture filtering. Then, the behavior of GPU framebuffer feedback loops can be studied with cellular automaton models. This is inspired by Hawick et al. [39] and Oliveira and Lotufo [73], whose work used

cellular automaton models for connected component labeling. Specifically, we are interested in a cellular automaton described by Algorithm 5.1.

Algorithm 5.1 Basic ID merging

```
For each cell
    A unique ID is assigned as the initial state;
For each iteration
    For each cell
        The cell's state is replaced by the maximum
        ID within its neighborhood;
```

We use a 1D example to demonstrate the reason that we chose this particular cellular automaton for examination of the random behavior of GPU framebuffer feedback loops. In Figure 5.1, a 1D cellular automaton has eight cells. At its initial state, each cell is assigned an integer ID, which is in ascending order. We examine three different methods of updating the cells. First, the cells are updated synchronously, which means we need an extra buffer to save the intermediate results. In this case, the order of how the cells are updated makes no difference to the results. It requires seven iterations to converge to all the same ID. Then, we update the cells asynchronously, i.e., reading and writing IDs without using an extra buffer. Since the order of how the cells are updated can be random and influence the result, we examine two extreme cases among all the combinations of update orders. The second method updates the cells asynchronously in order from left to right. The result looks exactly the same as when the cells are updated synchronously. Lastly, we update the cells asynchronously, but in reverse order. It only requires one iteration for the maximum ID to propagate.

We are able to make several observations from this example. First, if we color-map the IDs, we can see a fixed stripe pattern "marching" through the grid when the IDs are ordered and the updates are synchronous. Second, when the updates are not synchronous, the "marching" pattern is the same as synchronous updates only if the update order is the same as the ID order, and is disturbed

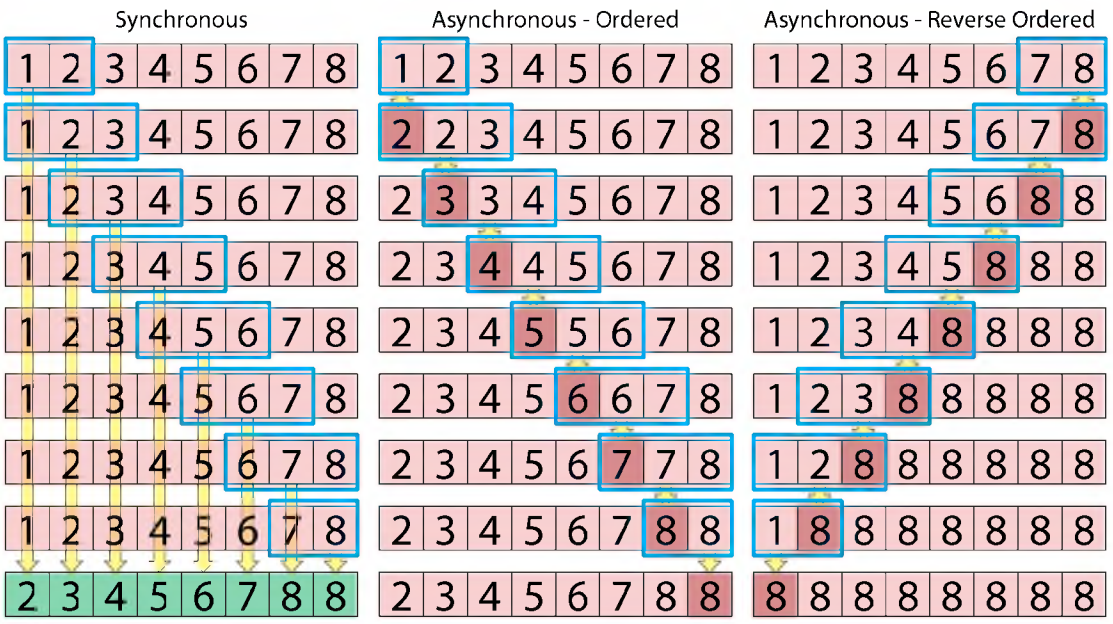

Figure 5.1. Comparison of synchronous and asynchronous updates of a 1D cellular automaton. The illustration shows only the first iteration in detailed steps. In the synchronous case, the original buffer is in red and the extra buffer for intermediate results is in green. Values are updated according to the maxima within the moving window (in blue), indicated by the yellow arrows. In the asynchronous cases, there is no extra buffer, so updates are immediate. The updated values in each step when the window moves are in dark red. We can repeat the iteration for the first two cases until all cells are updated to the maximum ID. Their results look exactly the same. The last case has already converged after the first iteration.

otherwise. We are able to detect such disorderliness by comparing the result from the asynchronous update to that of the synchronous update. Third, asynchronous update can potentially accelerate ID propagation.

The above example only shows the extreme or ideal cases. In reality, it would probably fall somewhere between the ideal cases. Suppose that the behavior of GPU framebuffer feedback loops is purely asynchronous and the chances of ordered and reverse ordered asynchronous updates are equal; we shall not see the regular patterns of "marching" IDs in the synchronous case, because they are so disturbed and become chaotic.

We extend the above idea and use it to detect the occurrence of asynchronous updates in GPU framebuffer feedback loops, which we assume to be the sole cause

of randomness in the process. For a given graphics card and a framebuffer texture of given size, we first assign IDs in ascending order. In 2D, this can be row-first or column-first, which does not influence the result. We run Algorithm 5.1 once in a framebuffer feedback loop and compare the outcome with that from a synchronous update. In the asynchronous result, we count the number of pixels that have different IDs than in the synchronous result. These pixels have different IDs because the orders of asynchronous updates are *against* the ID order. To count asynchronous updates of the reverse directions, we then do the same but with reverse ordered IDs. Both experiments are repeated for ten times. We then calculate the average percentages of asynchronous updates for both ordered and reverse ordered IDs. The two average values are added to estimate the total occurrence of asynchronous updates. We experimented with four graphics cards and each with eight different texture sizes. The results are illustrated in Figure 5.2.

The tested graphics cards should be representative for current (ca 2012) main-stream models from the two major GPU manufacturers: AMD and nVidia. The tested results reveal important characteristics of framebuffer feedback loops. Firstly, contrary to our initial speculation, framebuffer feedback loops are not entirely asynchronous. This is due to texture caching. Framebuffer feedback loop exhibits similar behavior to framebuffer Ping-Pong for small texture sizes (no asynchronous updates), as texture cache and graphics memory are working as two buffers for reading and writing. Secondly, occurrence of asynchronous updates increases as texture size increases. However, even for asynchronous updates, GPU threads tend to access memory with ascending order. This can be seen from the much higher occurrence rate of asynchronous updates when the IDs are in reverse order. Thirdly, GPUs from different manufacturers (AMD vs. nVidia) exhibit different occurrence rates of asynchronous updates. However, GPUs from the same manufacturer have similar results, even if they are in different series (for example, GeForce 400 series vs. 600 series). In conclusion, for specific graphics hardware, the behavior of framebuffer feedback loop is nondeterministic but predictable within a certain range, which is the result of a hybrid of synchronous

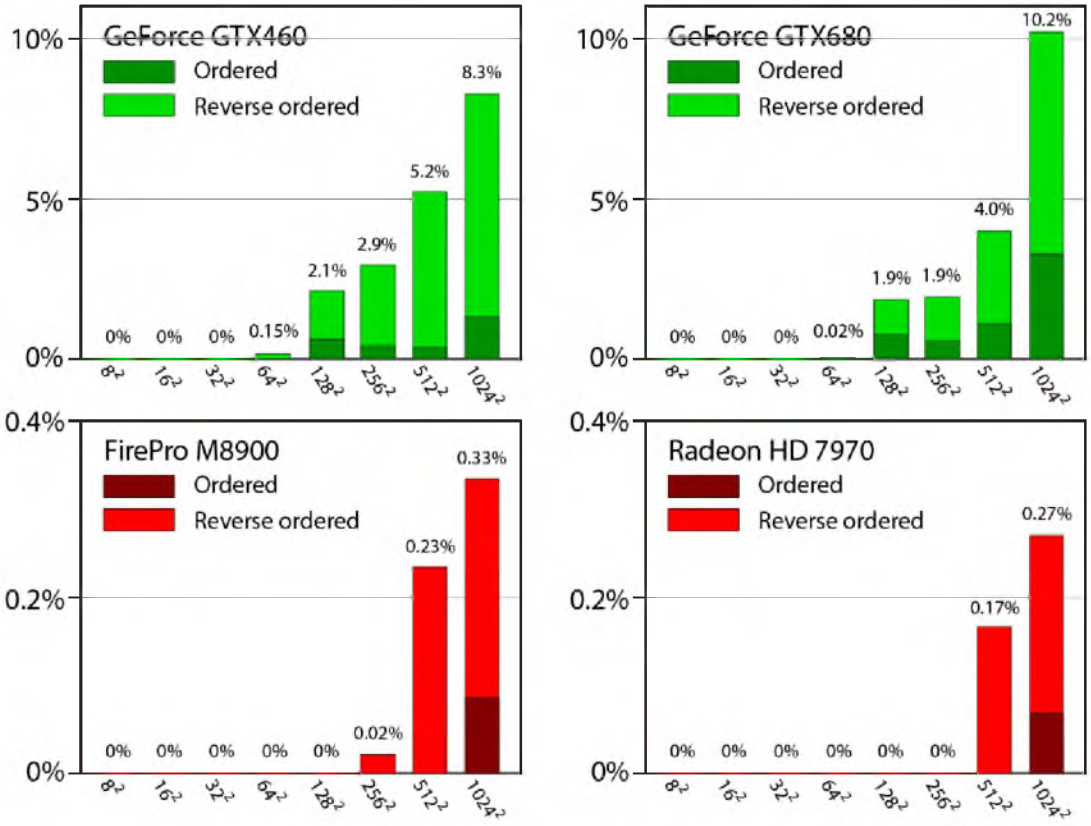

Figure 5.2. Test results of occurrence of asynchronous updates for four graphics cards: nVidia GeForce GTX 460, GTX 680, AMD FirePro M8900, and Radeon HD 7970. The horizontal axes of all plots are texture sizes tested. The vertical axes are the occurrence of asynchronous updates in percentage.

and asynchronous updates. It is worth mentioning that the authors did the tests within their available resources. The behavior of framebuffer feedback loop may vary greatly for older or future hardware, or integrated GPUs. We continue our discussion in the following sections with regard to the tested graphics hardware. Framebuffer feedback loops are always used in the subsequent discussions on Synthetic Brainbows.

5.2 Synthetic Brainbows

5.2.1 ID Shuffling

In fact, Algorithm 5.1 was discussed in the work of Hawick et al. [39] and that of Oliveira and Lotufo [73] as a local merging step in their GPU implementations

of connected component labeling. If we introduce a binary mask into an iterative process, assigning and propagating IDs only within the masked regions, it converges to the labeled connected components of the mask. With framebuffer feedback loops, the result still converges and is deterministic, since cell values are monotonically increasing and upper bounded within each connected component. This also can be considered as a simple case of chaotic relaxation [17]. However, if we focus on the process itself rather than its convergence, we should be able to observe one problem when IDs are ordered as initial states. For example, a spiral is used as the binary mask in Figure 5.3. If we consider the spiral as a complex structure, the visualization task here is to decompose the structure into simpler shapes so that each component as well as the spatial relationships among components can be more easily studied. Connected component labeling, which considers the complex structure as one component, does not fulfill the requirement. A simple solution is to stop the iterative process of local ID merging at fixed iterations. This generates stochastic patterns due to asynchronous updates. But also, because the tested graphics cards largely exhibit synchronous behavior, as discussed in Section 5.1, local ID merging generates many small subregions (Figure 5.3 (a) and (b)). The high frequency patterns in these unmerged regions can be visually distractive. To compensate for this and generate fewer subregions after certain iterations, we propose ID shuffling. ID shuffling does not randomly change the order of IDs. Instead, IDs are meant to be *evenly* distributed. This is formally defined as maximizing the grid distance between any ordered pair of adjacent IDs. Intuitively, it can be understood as the process of placing IDs, from large to small, onto an empty grid. Wherever an ID is placed on the grid, the next smaller ID is placed so that the pair can be as far apart as possible. The result is that local merging is centered at local maxima and small subregions are quickly merged into larger regions (Figure 5.3(c)).

For easy implementation, we first consider an ID shuffling algorithm for grids of size 2^n. The shuffled IDs are easily generated from their grid coordinates, as the IDs are spatially placed according to a binary (or quad- for 2D, or oct-

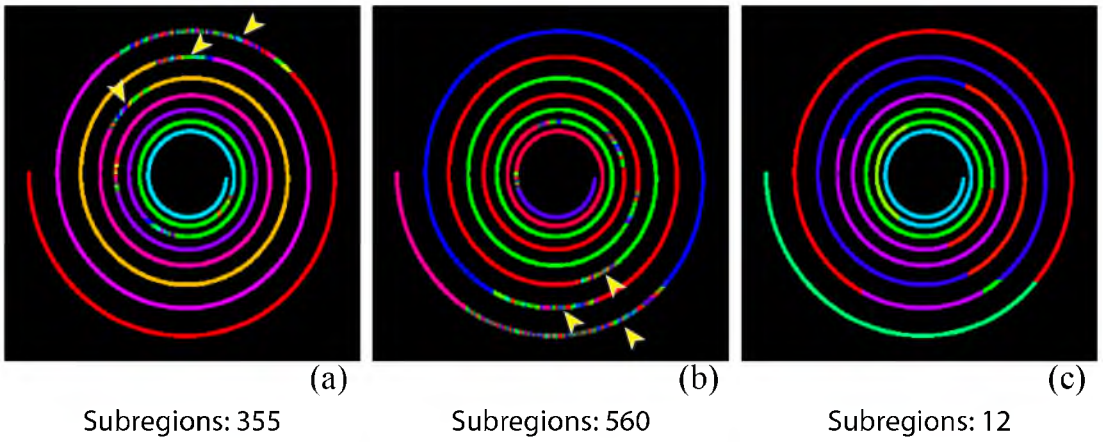

Figure 5.3. A 512×512 binary image of a spiral is used as a mask for Algorithm 5.1. After 512 iterations, different ID orderings exhibit different patterns when IDs are color-mapped. (a) IDs are in ascending order. (b) IDs are in descending order. (c) IDs are shuffled with Algorithm 5.3. The numbers of remaining subregions are shown below the images. Yellow arrowheads point to regions where high amount of unmerged subregions are present, due to the largely synchronous behavior of the graphics card. With shuffled IDs, the algorithm is able to generate fewer subregions with the same number of iterations. The tests are done on an AMD Radeon HD 7970 graphics card, which exhibits the least asynchronous behavior in Figure 5.2. There are fewer remaining subregions when a more asynchronous graphics card is used, e.g., nVidia cards. However, the difference between ordered and shuffled ID orderings is still quite large.

for 3D, etc.) encoding tree. We first introduce the shuffling algorithm for a 1D grid (Algorithm 5.2), which is illustrated in Figure 5.4. In the algorithm, the function reverse_bit() reverses the order of the binary code of the input integer. The subtraction step at the end is only for indices starting from 0. We want to exclude 0 from valid IDs, since it is used as a mask value. It is equivalent to increasing the reversed index value by 1.

Shuffling algorithms for higher dimensional grids are extensions of Algorithm 5.2. In Algorithm 5.3, the function interleave_bit() combines the bits of its multiple inputs in an interleaving fashion. For example, in a 3D grid, we have calculated I'_x, I'_y, and I'_z for one cell. The first bit of the function's output I' is the first bit of I'_x, and then the second bit of I' is the first bit of I'_y, and then the first bit of I'_z, and so on.

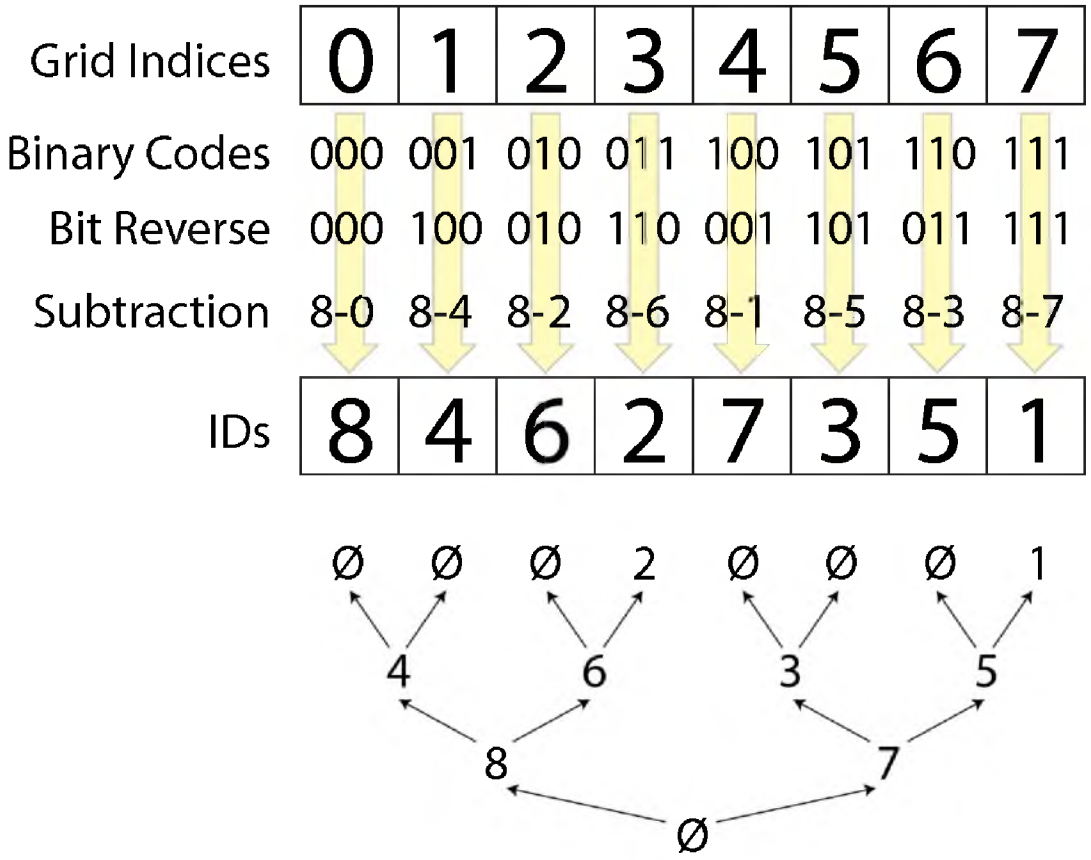

Figure 5.4. ID shuffling for an 8-cell 1D grid. The binary code of each cell index is reversed and then subtracted from the total cell number. The IDs are placed in the order of visiting the binary tree shown below, with depth-first traversal. In the tree, larger IDs are on nodes of higher levels.

Algorithm 5.2 ID shuffling for a 1D grid

Define N = the number of cells of the grid and
N = 2^n, ($n \in Z$);
For each cell
    I = the cell's grid index;
    I' = reverse_bit(I);
    ID = N - I';

For a grid of an arbitrary size, it is considered as a subregion of a larger grid of size 2^n. Then, IDs of any grid can be calculated using Algorithm 5.3. Our

Algorithm 5.3 ID shuffling for a high-dimensional grid

Define $\{N_i\}$ = the number of cells in each dimension
and $N_i = 2^{n_i}, (n_i \in Z)$;
Define N = $\prod(N_i)$;
For each cell
    $\{I_i\}$ = the cell's grid index in each dimension;
    $\{I'_i\}$ = {reverse_bit(I_i)};
    I' = interleave_bit($\{I'_i\}$);
    ID = N - I';

algorithm is not the unique way to shuffle IDs. Equivalent algorithms can be derived, for example, by swapping nodes on the same level of the binary tree in Figure 5.4. However, our algorithm is suitable for parallel evaluation, as each cell's ID is uniquely defined by its indices. For repetitive evaluations of different grids, IDs can be precalculated and reused. Furthermore, in addition to asynchronous computing models, ID shuffling is potentially an improvement to existing connected component labeling algorithms, since it accelerates the local merging step. However, for visualization purposes, our ID merging algorithm with shuffling can only be applied on binary data. The decomposition of complex structures still seems to be arbitrary. In order to apply it on grayscale data and for the colorization to follow structural information, finer control over the local merging process is necessary, which is discussed next.

5.2.2 Monte-Carlo Sampling

Our goal here is to apply ID merging on a single channel of confocal microscopy data, and generate Brainbow-like images. We want the colorization process to be applied directly as we do not want to rely on presegmented results. Since we want to have unique IDs for individual regions, one difficulty of applying the colorization described in Section 5.2.1 to grayscale data is that IDs cannot be simply scaled according to scalar intensities. However, we can control the merging speed of IDs based on scalar intensities. This is achieved through temporal Monte-Carlo

Algorithm 5.4 Colorization of a grayscale volume

```
Define S the scalar volume of original data;
Define ID the ID volume generated by Algorithm 5.3;
For each cell in ID
    M = the measure of features in S;
    N = a sample from a 4D noise function;
    If M > N
        The cell's ID is replaced by the maximum ID
        within its neighborhood;
```

sampling. The algorithm for colorizing a grayscale volume is listed.

In Algorithm 5.4, M is a scalar value calculated from the original scalar volume. It measures the features that we use to control the merging speed. For example, the stopping function in a standard anisotropic diffusion [80] can be used as the measure of homogeneity. We can use this measure if we want the merging to be faster in homogeneous regions and slower at edges (less homogeneous). The value N can be seen as a pseudo-random number generated from a 4D (3D plus time or iteration) noise function [37]. If the value of M is higher, there is also a higher chance that the ID is merged, and vice versa.

We first experimented with a common measure of edges, which is defined by the gradient magnitude of the intensity value S:

$$M = e^{-\frac{|\nabla(S)|^2}{\sigma^2}} \tag{5.1}$$

Figure 5.5 shows the result of applying Algorithm 5.4 on a confocal scan (512×512×85×8bit) of a *Drosophila* brain. The nervous system of the *Drosophila* brain has complex structures, where important structures can easily be obscured. We ran 200 iterations of ID merging and examined the patterns generated from Algorithm 5.4. Notice that the ordered ID sequence was repeatedly mapped to a palette of bright colors that resemble fluorescent markers. Because of ID shuffling (Algorithm 5.3), the colors were spatially shuffled too. The colored volume was then modulated by the original scalar intensities. The number of iterations was

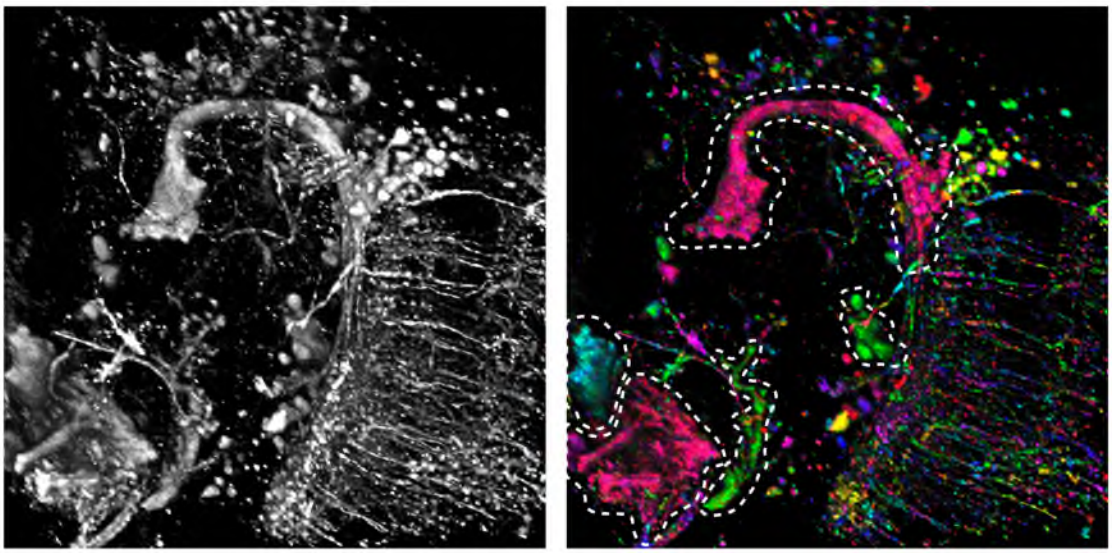

Figure 5.5. Colorization of a confocal scan of a Drosophila brain (512×512×85×8bit). (a) The volume rendering of the original dataset. (b) The volume rendering of the colorized dataset. Dotted outlines indicate large and homogeneous structures. It took 200 iterations to generate the result. Generating the result took around 1 second on an AMD Radeon HD 7970.

chosen to generate the desired color variations. Comparing the colorized volume (Figure 5.5 (b)) with the original volume (Figure 5.5 (b)), we can observe that large and homogeneous structures are emphasized, which are indicated by the dotted outlines in Figure 5.5 (b). Small structures are also distinctively colored. This is helpful when one structure is obstructing another. Their spatial relationship becomes clear, as they are colored differently. An apparent drawback of using only gradient magnitude as the measure for edges is that faintly connected structures, such as the fibers in the lower right region of Figure 5.5, are colored differently even for the same branch. To generate the result in Figure 5.5, σ in Equation 5.1 was set to 0.5. If we further increase its value, more structures are merged and colored the same, which reduces the resolving capability. This is illustrated in Figure 5.6 by an example of two idealized touching biological cells, where we have to increase the σ value in order to merge the surrounding IDs. For just two cells, we can easily stop the merging process when the two groups of IDs

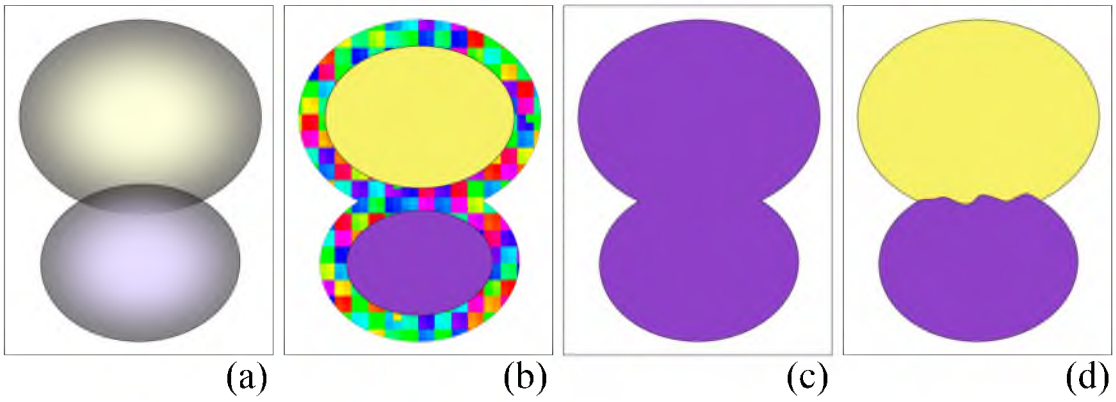

Figure 5.6. An illustrated example of two idealized touching biological cells. (a) The original data. Cells have high intensity and low gradient magnitude at their centers and low intensity but high gradient magnitude at borders. (b) Algorithm 5.4 is used with the measure in Equation 5.1. When the σ value in Equation 5.1 is low, the centers of the cells are colored differently. However, they are surrounded by a cloud of various colors (IDs), which obscures the content inside. (c) When we increase the σ value, the two cells are fused together. (d) When we introduce a size constraint, the two cells can be colored as desired.

join at the boundary, similar to Figure 5.6 (d). This becomes impractical for a large amount of cells or complex structures, since the iteration number is a global parameter and cannot be tuned for all structures of different sizes.

A great advantage of using the ID merging process is that the size of each individual structure having the same ID can be retrieved by counting the number of cells with the same ID. The size of each structure is then used to control the merging process. The problem of two touching cells can be solved using a two-pass method. In the first pass, a low σ value is used and the result looks like Figure 5.6 (b). Then, we count the size of each structure having the same ID, similar to calculating component sizes in connected component analysis. In the second pass, we use a high σ value and set a constraint on component size, which is described in Algorithm 5.5.

If the size constraint is set to lower than the size of the smaller cell, the surrounding IDs in Figure 5.6 (b) are merged into the two cells. The IDs from the two cells, however, cannot be merged into one another, because of the size constraint. The result should look like Figure 5.6 (d). Notice that although the size

Algorithm 5.5 Synthetic Brainbow (colorization of a grayscale volume with size constraint)

```
Define S the scalar volume of original data;
Define ID the ID volume generated by Algorithm 5.3;
Define B a Boolean volume whose voxels represent
if they are in a component with size over a threshold;
For each cell in ID
    M = the measure of features in S;
    N = a sample from a 4D noise function;
    If M > N and !B
        The cell's ID is replaced by the maximum ID
        within its neighborhood;
```

constraint is a global parameter, it is used to safely merge IDs in noisy regions. So, it is usually set at a relatively small value. Larger structures are merged by adjusting the measure of features and iteration numbers.

5.2.3 Results and User Survey

Using the above method, we generated Synthetic Brainbows for three single-channel confocal scans. We were able to adjust the number of iterations, the σ value in Equation 5.1, and the size constraint during the ID merging process. For each scan, we performed several experiments with different parameter combinations until reaching a satisfactory result. The results are listed in Figures 5.7, 5.8, and 5.9. These results are from an AMD Radeon HD 7970 graphics card.

For most cases, e.g. Figure 5.7 and 5.8, our Synthetic Brainbow technique does not alter information from original datasets. It enhances visualization by randomly applying colors to different structures. We chose bright colors that resemble fluorescent markers used in biology research, so that structures are not accidentally emphasized/de-emphasized because of color variance. For highly noisy datasets, e.g., Figure 5.9, we suppressed noise with MIP. We believe such colorization can help visualizing complex structures in biology research. Since the colorization process takes relatively short time, it can be integrated into a

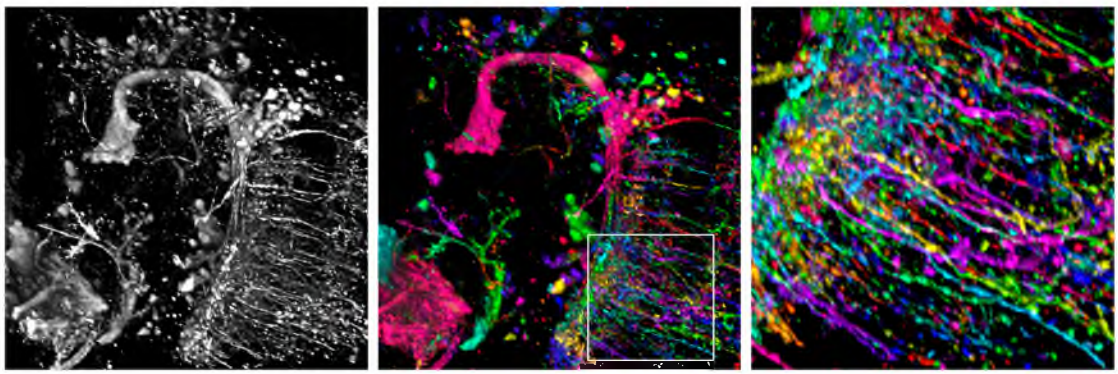

Figure 5.7. Synthesized Brainbow of the same confocal scan of a Drosophila brain (512×512×85×8bit) in Figure 5.5. (a) The volume rendering of the original dataset. (b) Rendering of the colorized result generated with size constraint. (c) A close-up of the fibers. Individual fibers are connected and different fibers are colored differently. In the first pass of ID merging, iteration number is set to 200, and σ is set to 0.5. In the second pass of ID merging, iteration number is set to 200, σ to 1.0, and size constraint is set to 100 voxels. The colorization process took 2.81 seconds on an AMD Radeon HD 7970.

biologist's visualization workflow. Another prospective use is when a group of biologists are looking at scans, instead of pointing on the datasets and referring structures as "this" or "that", specific color names can be used. However, this can only be validated when the Synthetic Brainbows are used in practice.

To address the concern that results from the Synthetic Brainbows may vary from one graphics card to another, we also generated Synthetic Brainbow images with other graphics cards tested in Section 5.1. The results are in Figure 5.10, 5.11, and 5.12. Because occurrence of asynchronous updates is about 5% at the highest for 512x512 textures, only limited variation can be observed from these results. However, since a comprehensive test of all graphics card models is impossible at current stage, consistent results can only be guaranteed on the tested models.

Since biologists are the potential users of our technique, we created an online survey and sent its link to biologists that are experts in confocal microscopy. The participants should be familiar with the Brainbow technique, but they may not necessarily be working with the technique. There were two parts in the survey. In each question of the first part, an image was shown and participants

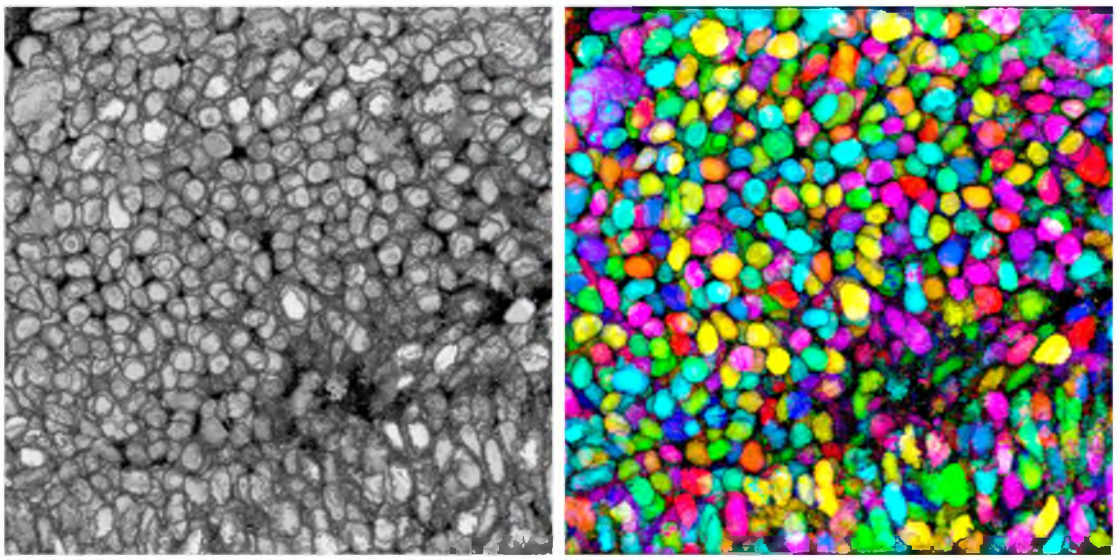

Figure 5.8. Synthesized Brainbow of a confocal scan of an eye of a zebrafish embryo (512×512×33×8bit). (a) The volume rendering of the original dataset. (b) The volume rendering of the colorized result. In the original scan, many structures are fused together, which are better discriminated in the colorized result. The colorization took two passes. In the first pass, iteration number is set to 50, and σ is set to 0.35. In the second pass, iteration number is set to 300, σ to 1.0, and size constraint is set to 250. The colorization process took 1.34 seconds on an AMD Radeon HD 7970.

were asked how likely the image was generated by the Brainbow technique. There were eight images: four generated with our technique, two true Brainbow images, and two images generated by first anisotropic diffusion [80] and then thresholding plus connected component labeling. The images were shown in random order. In the second part, we revealed the images that were generated by our technique and showed their original renderings as in Figure 5.7, 5.8, and 5.9. We asked the participants how likely they would use this technique for visualization enhancement in their research. We received answers from 16 participants, some of which also left comments. The answers to the first part are plotted in Figure 5.13. Most participants agreed that our technique was able to generate results similar to Brainbow. Furthermore, 70% of the participants showed great interest in using our technique. From the survey results and participants'

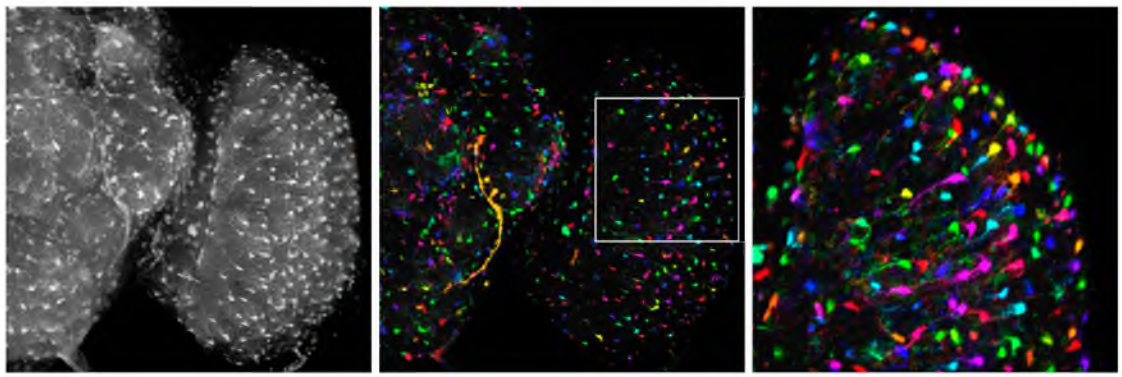

Figure 5.9. Synthesized Brainbow of a confocal scan of a Drosophila brain (512×512×115×8bit). (a) The volume rendering of the original dataset. This is a highly noisy dataset. (b) The volume rendering of the colorized result. The colored volume is rendered with maximum intensity projection (MIP) [109] plus a shading overlay (Section 3.6, Chapter 3), in order to see the colored structures clearly. (c) A close-up of the cells. The colorization took two passes. In the first pass, iteration number is set to 200, and σ is set to 0.35. In the second pass, iteration number is set to 10, σ to 1.0, and size constraint is set to 50. This 10-iteration process is then repeated five times in the second pass. This is because the dataset is highly noisy and we need to look at the result and decide if more iterations are necessary. The colorization process took 1.98 seconds on an AMD Radeon HD 7970, excluding the time for manual parameter adjustment.

feedback, we learned that visualizations altering the appearance of the original data would be commonly rejected. Researchers often want fine details and sometimes even noise to be preserved. Interestingly, some of the biologists were able to tell that data had been altered because of their over-smoothed and thus unnatural look. This is the main reason why the thresholded and labeled results had low scores. The data in Figure 5.8 had been preprocessed with a median filter. We believe this led to its lower score (25%, the second in Figure 5.13). Thanks to the ID merging process, our method is able to preserve fine details of the original data, which is important to biologists for a visualization technique.

ID merging with asynchronous cellular automata had been used in connected component analysis, which we regard as the primitive form of more sophisticated colorization. Our quest for computationally generating Brainbow-like images from single-channel confocal microscopy data started with examinations of

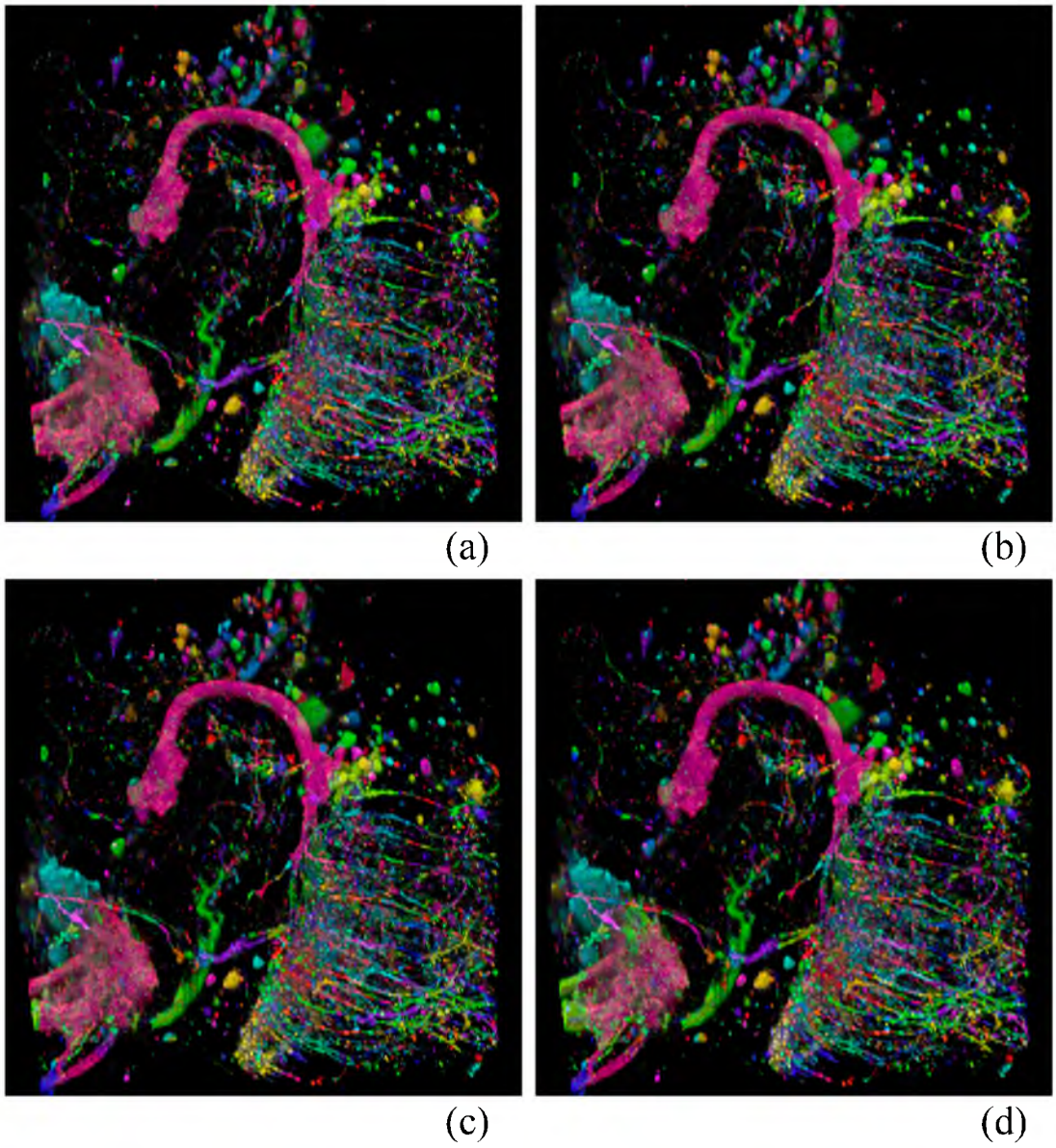

Figure 5.10. Synthesized Brainbow of the same confocal scan of a Drosophila brain (512×512×85×8bit) in Figure 5.5, using different graphics cards. (a) Result from nVidia GeForce GTX 460. (b) Result from nVidia GeForce GTX 680. (c) Result from AMD FirePro M8900. (d) Result from AMD Radeon HD7970.

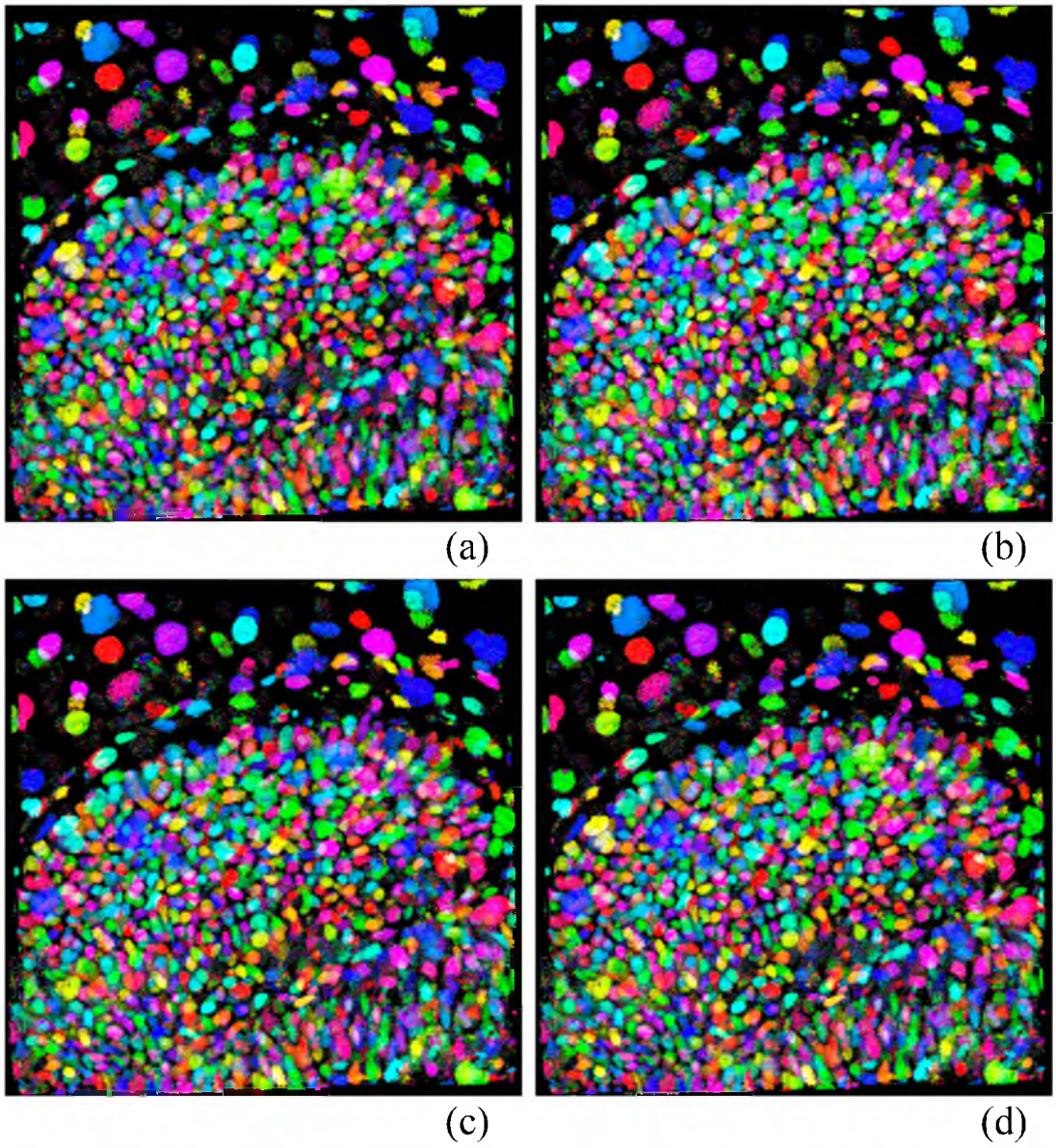

Figure 5.11. Synthesized Brainbow of the same confocal scan of an eye of a zebrafish embryo (512×512×33×8bit) in Figure 5.8, using different graphics cards. (a) Result from nVidia GeForce GTX 460. (b) Result from nVidia GeForce GTX 680. (c) Result from AMD FirePro M8900. (d) Result from AMD Radeon HD7970.

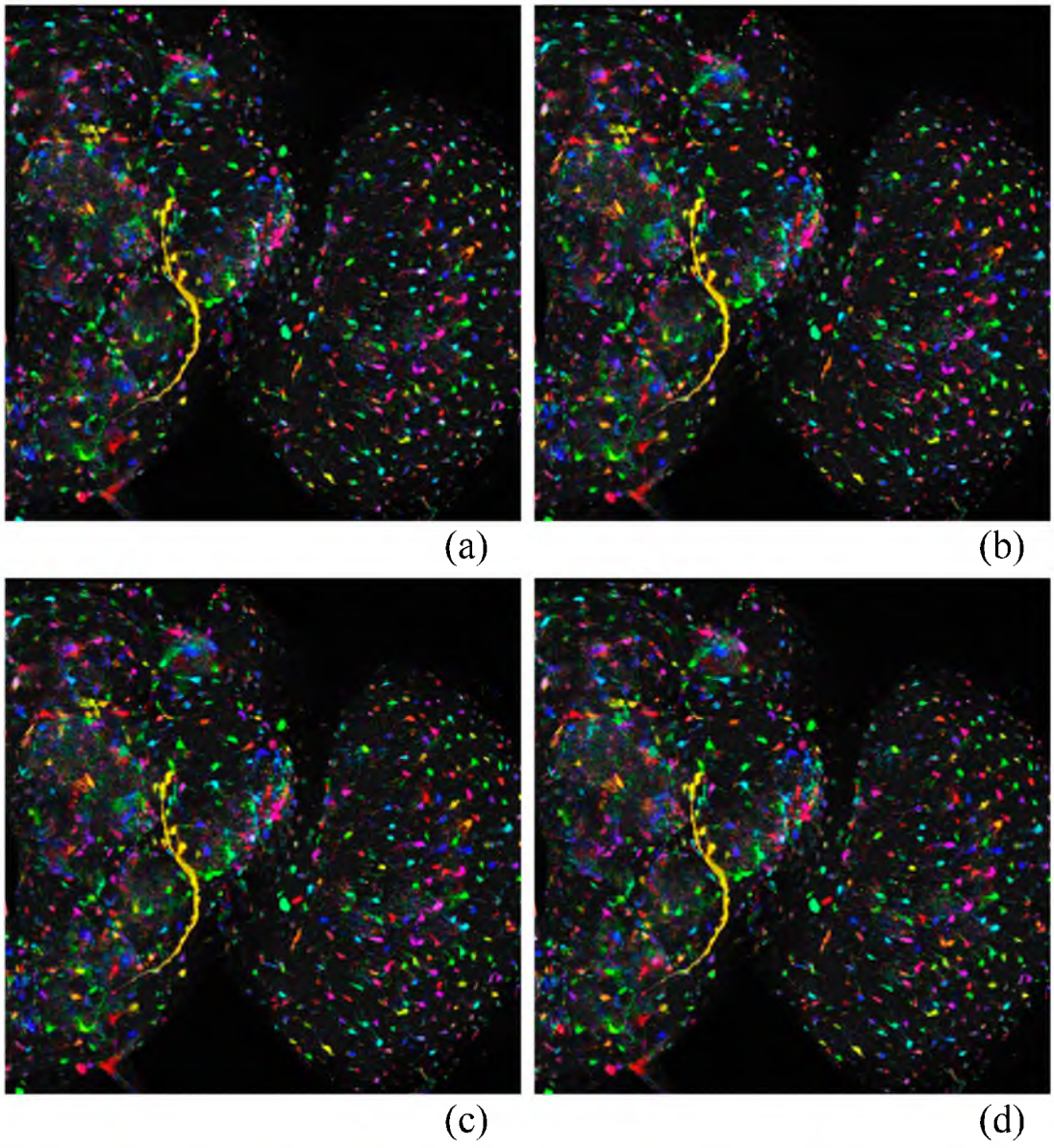

Figure 5.12. Synthesized Brainbow of the same confocal scan of a Drosophila brain (512×512×115×8bit) in Figure 5.9, using different graphics cards. (a) Result from nVidia GeForce GTX 460. (b) Result from nVidia GeForce GTX 680. (c) Result from AMD FirePro M8900. (d) Result from AMD Radeon HD7970.

Figure 5.13. Results from the first part of our survey. The collective answers to the likelihood of each image being generated with the Brainbow technique are plotted in one bar plot. The length of a bar represents the frequency of each choice being selected (5 - most likely, 1 - most unlikely). Here, the images are grouped according to techniques used. In the survey, they were shown to the participants in random order. Above the plots are the combined percentages of the participants who answered 5 or 4. The answers to most of our images are similar to those of the true Brainbows.

GPU framebuffer feedback loops. We initially thought it would be purely nondeterministic because of asynchronous memory access in parallel. However, when we tested its behavior with a cellular automaton, we found the behavior of GPU framebuffer feedback loop, despite being nondeterministic, is less random than we thought. In order to use the patterns generated in the iterative process of GPU framebuffer feedback loops to synthesize Brainbow-like results, we introduced ID shuffling and Monte-Carlo sampling into the ID merging process. Our technique is able to enhance visualizations of data with complex structures, such as the biological datasets demonstrated in this chapter. Our technique has advantages over traditional segmentation plus labeling methods because of its speed, and also because it preserves fine details of original data. Both make it a visualization technique appealing to domain experts, as we learned from a user survey. Currently, a drawback of our technique is the lack of an intuitive user interface, which limits its users to the authors and their close collaborators. We would like to address this issue in future work.

There are apparent similarities between our technique and many segmentation methods for cellular or fibrous structures in biological data. Algorithm 5.5 can be considered as a general framework for segmentation, instead of a specific segmentation method. Firstly, it combines component labeling and feature detection into one process. It is also related to fuzzy connected component analysis [101]. An important benefit of such combination is that size information is readily available and used as a constraint. The size calculated from component labeling is a more accurate descriptor than the previously proposed size-based transfer function [22]. Our size descriptor works better with fibrous and branching structures, since they may have big size but be considered small by a local size descriptor. Secondly, the quality of segmentation can be refined by using more specific feature measures. For example, a measure of tubeness [90] can be added for fibers, and a measure using similarity between smoothed gradient vectors [60] can be added for cells. Thirdly, Algorithm 5.5 can be used with or without GPU framebuffer feedback loops. The purpose of this chapter is to find a random colorization technique that can be used with GPU framebuffer feedback loops, which saves memory and utilizes memory bandwidth more efficiently. We paid more attention to the stochastic patterns generated in the process and did not discuss convergence in great detail. However, when appropriate constraints are applied, the process can still converge to a segmentation of the input.

5.3 Tracking Cells

Biologists often want to find out the trajectories of cell movements in time sequences acquired by confocal microscopy. Individual cells have to be distinguished not only within one time frame, but also through multiple frames. Tracking individual cells automatically becomes problematic, as cells in proximity are often densely packed and have similar shapes. An automatic tracker can be very easily confused and generate unreliable results. By leveraging the parallel processing power of GPUs, we propose a cell tracking method using our Synthetic Brainbow technique (Section 5.2). In general, it is easier for an automatic tracker

to identify cells through time, if each individual cell has been identified within each time frame. A bipartite matching algorithm [112] can be used to calculate the best match for the already identified cells between two consecutive time frames. Application of the Synthetic Brainbows to cell tracking is relatively straightforward: we first generate synthesized Brainbows for each time frame, where cells have unique IDs; then, we calculate the maximum bipartite matching between consecutive time frames, where matched IDs are unified into the same ID.

5.3.1 Synthetic Brainbows for Cells

Decreasing brightness and signal-to-noise ratio through time are challenges for cell tracking with confocal microscopy time sequences. To obtain consistent results through time frames, we improved the measure in Equation 5.1 specifically for cells in confocal microscopy data. Similar to multidimensional transfer functions [53], our measure is composed of four submeasures. The combination of these submeasures aims for a better definition of cell boundaries during ID propagation.

• Scalar value. The measure is calculated similarly as in Equation 4.9.

$$m_1(V) = \begin{cases} e^{-\frac{(V-t_1)^2}{k_1^2}} & V < t_1 \\ 1 & otherwise \end{cases} \tag{5.2}$$

In Equation 5.2, k_1 and t_1 are two user adjustable parameters for the shape of the Gaussian falloff. In confocal microscopy data, the center of a cell has high scalar values, which decrease towards the boundary. Therefore, ID propagation is faster at the center of a cell than its boundaries.

• Scalar variance. We calculate the absolute deviation $var(V)$ of the scalar values within a neighborhood. Then, the measure is calculated from the absolute deviation.

$$var(V) = \frac{1}{n}\sum_{i=1}^{n}\left|V_i - \overline{V}\right|$$
$$m_2(V) = e^{-\frac{var(V)^2}{k_2^2}} \tag{5.3}$$

In Equation 5.3, $\overline{V}$ is the average of the scalar values within the neighborhood and k_2 is a user adjustable parameter. In confocal microscopy data, the center of a cell usually has less scalar value variation than its boundary, which can become quite noisy (high scalar value variation) between two cells. The neighborhood stencil is usually anisotropic. We use a voxelized ellipsoid that has 5×5 voxels on the XY plane and 3 voxels along the Z axis. This is because most confocal data are anisotropic and we want neighboring voxels along the Z axis to have less influence. The result is that ID propagation is faster at smooth regions than noisy regions.

• Gradient magnitude. The measure is calculated similarly as in Equation 4.9. It is calculated as the Gaussian of the gradient magnitude, which is commonly used to measure boundaries.

$$m_3(V) = \begin{cases} 1 & |\nabla V| < t_3 \\ e^{-\frac{(|\nabla V| - t_3)^2}{k_3^2}} & otherwise \end{cases} \tag{5.4}$$

In Equation 5.4, k_3 and t_3 are two user adjustable parameters. This is the common measure for boundaries used in Section 5.2.2.

• Gradient direction variance. We first calculate the normalized average gradient vector within a neighborhood. Then, we calculate the averaged dot product between each normalized gradient vector and the normalized average gradient vector. Each dot product is weighted by the magnitude of each gradient vector, so that gradient vectors with less magnitude have less influence on the result. The result is used to calculate the fourth measure of gradient direction variance.

$$\overline{\nabla V} = normalize(\frac{1}{n}\sum_{i=1}^{n} \nabla V_i)$$
$$gvar(V) = \frac{1}{n}\sum_{i=1}^{n} |\nabla V_i| dot(\overline{\nabla V}, normalize(\nabla V_i)) \quad (5.5)$$
$$m_4(V) = e^{-\frac{gvar(V)^2}{k_4^2}}$$

In Equation 5.5, $\overline{\nabla V}$ is the normalized average gradient vector; $gvar(V)$ is the gradient direction variance; k_4 is a user adjustable parameter. This measure helps ID propagation stop at regions between cells, since gradient vector changes direction at these regions, thus high gradient direction variance. Notice that gradient direction also changes at the center of a cell in theory. This is why we want to weight the dot product with the gradient magnitude, as the center usually has low gradient magnitude, which is used to decrease the center voxels' influence on the gradient direction variance. The same neighborhood as in the calculation of the scalar variance measure is used here, to lessen the influence of neighboring voxels along the Z axis.

As mentioned in the beginning of Section 5.3.1, the method looks similar to multidimensional transfer functions, which would be four-dimensional in this case. However, finding proper parameters for above equations does not seem to be as difficult as in a four-dimensional transfer function. This is because the influences of the measures are on the ID propagation speed, instead of directly on rendering properties (color and transparency) of the voxels as in multidimensional transfer functions. Using the Synthetic Brainbow technique, the final colors of the voxels are generated through an iterative process by accumulating the influences of the above measures. The final results become less sensitive to parameter changes. More importantly, results from different time frames in a time sequence confocal data become less sensitive to parameter changes. Thus, it is possible to use the same set of parameters for above measures through all the time frames. This becomes important for time sequences with many frames. We are able to

determine the parameters according to just one or two and apply them to all frames, instead of adjusting for each frame.

Figure 5.14 shows the results of applying the Synthetic Brainbow technique with above measures. For all time frames of the confocal sequence, the same parameters were used in two passes. In the first pass, the settings were: $t_1 = 1.0$, $k_1 = 0.01$, $k_2 = 0.07$, $t_3 = 0.0$, $k_3 = 0.04$, $k_4 = 0.07$, *iterations* = 50, size constraint ignored; in the second pass, the settings were: $t_1 = 1.0$, $k_1 = 0.1$, $k_2 = 0.7$, $t_3 = 0.0$, $k_3 = 0.4$, $k_4 = 0.7$, *iterations* = 15, size constraint set to 30.

5.3.2 ID Matching

It is easy to observe that sometimes, the same cell does not have the same color in different time frames in Figure 5.14. We need to match IDs in different time frames. This is accomplished using a standard algorithm in graph theory: weighted maximum bipartite matching [112]. Here, we consider each individual ID as a node on a graph. If two IDs from two consecutive time frames have overlapped voxels (voxels occupy the same space coordinates, but in different time), an edge is connected between them. The edge is weighted by the number

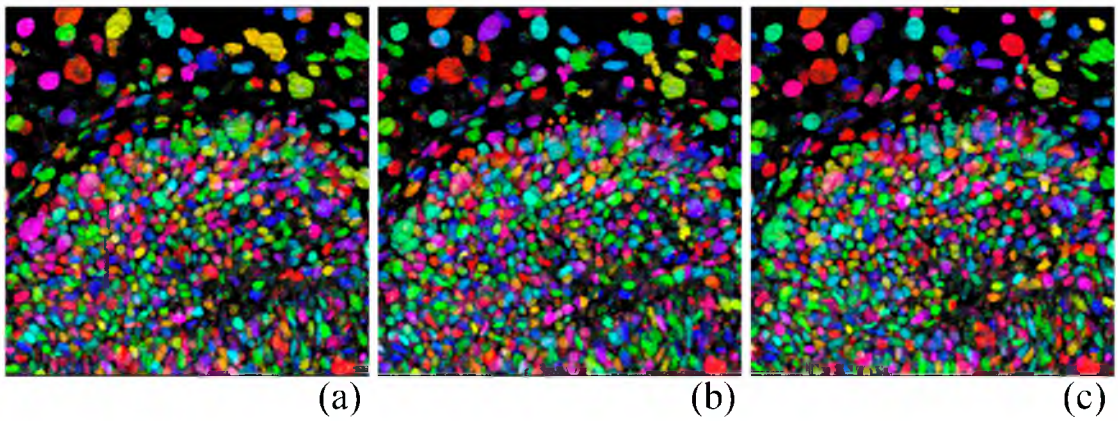

Figure 5.14. Results of applying the Synthetic Brainbow technique on a time sequence confocal data. Three consecutive time frames are shown here ((a), (b), and (c)). Two passes of ID propagation are used. In the first pass, the settings were: $t_1 = 1.0$, $k_1 = 0.01$, $k_2 = 0.07$, $t_3 = 0.0$, $k_3 = 0.04$, $k_4 = 0.07$, *iterations* = 50, size constraint ignored; in the second pass, the settings were: $t_1 = 1.0$, $k_1 = 0.1$, $k_2 = 0.7$, $t_3 = 0.0$, $k_3 = 0.4$, $k_4 = 0.7$, *iterations* = 15, size constraint set to 30.

of overlapping voxels. The graph is bipartite because an overlap (an edge) can only occur between IDs belonging to different time frames. Figure 5.15 illustrates the construction of such a graph.

The weighted maximum bipartite matching algorithm can be found in literature on graph theory [112]. A detailed discussion is out of the scope of our work. The algorithm is listed in Algorithm 5.6.

Algorithm 5.6 Weighted maximum bipartite matching.

```
Start with a bipartite graph with no matched edges established;
Repeat
    For all unmatched nodes on one side of the bipartite graph
        Breadth-first search and find an augmenting path with the maximum
        weight;
    If found the augmenting path and its weight > 0
        Flip the edges;
    Else
        Stop. The current matching is the result.
```

In Algorithm 5.6, a path on the bipartite graph can have both matched and unmatched edges. A path that has alternating matched and unmatched edges is called an alternating path. An alternating path whose total weight of unmatched edges is greater than that of matched edges is called an augmenting path. The weight of an augmenting path is calculated by subtracting the total weight of matched edges from that of unmatched edges. Flipping an alternating path means marking matched edges as unmatched and unmatched as matched. This algorithm is applied to a bipartite graph generated from two consecutive time frames that have been processed with the Synthetic Brainbow technique. Then, the entire time sequence is processed with this method. Figure 5.16 shows the results after ID matching for the confocal dataset in Figure 5.14.

Synthetic Brainbows provide us a simple and easy-to-implement method to track cells in time sequence confocal microscopy data. By adopting the maximum matching algorithm, a global optimal match can be guaranteed between two

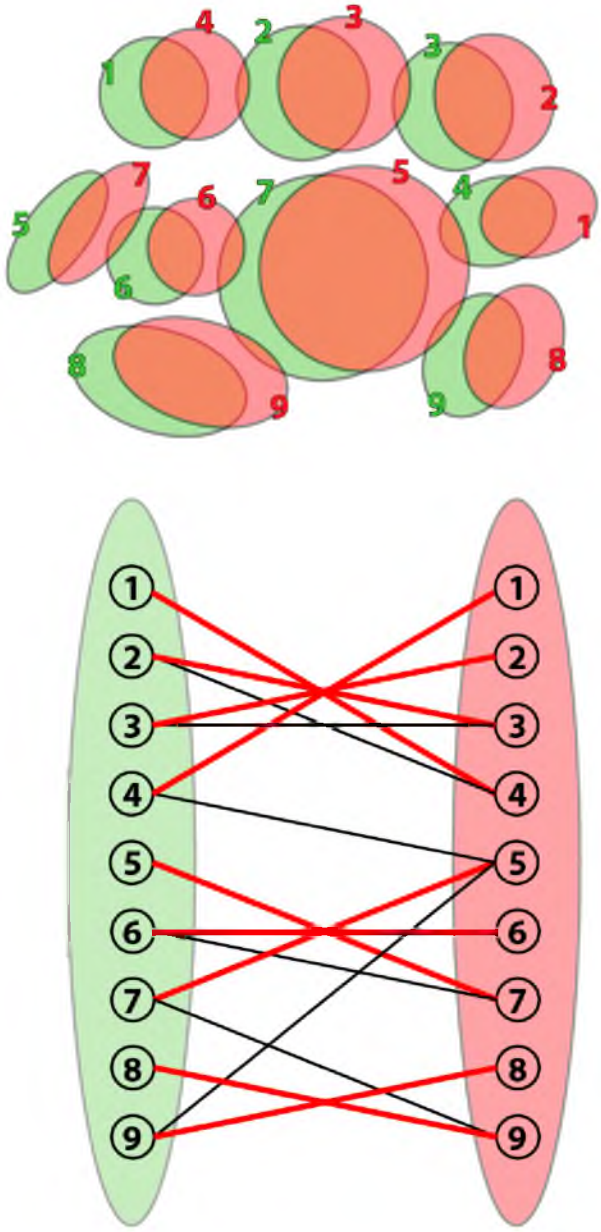

Figure 5.15. Conversion from cells with IDs to a bipartite graph. The green cells and red cells are from two consecutive time frames. They are super-imposed in the upper part of the figure. Each cell has an unique ID in each frame, but the same cell may not have the same ID from two frames. We can then create a graph in the lower part according to their overlapping. Each edge in the graph represents an overlap and is weighted according to the overlapped area (not shown). The red edges are the matched edges after applying the weighted maximum matching.

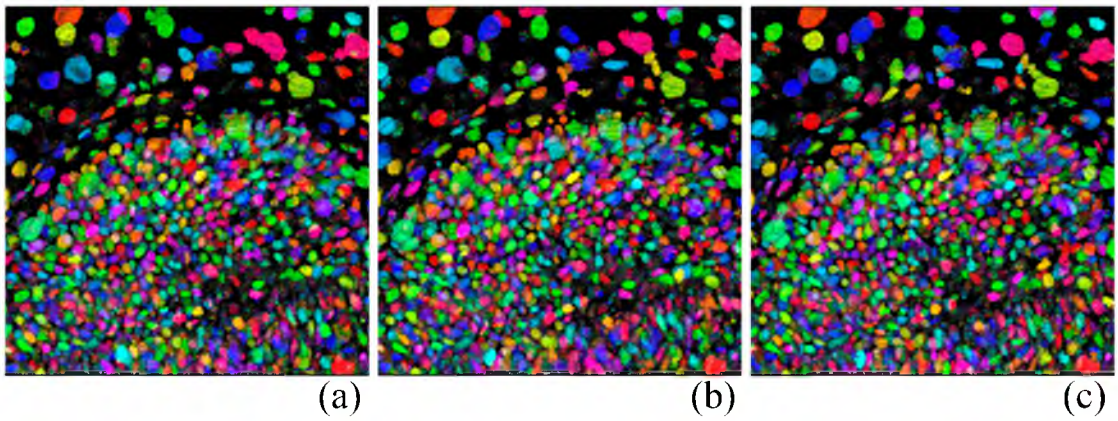

Figure 5.16. Results of ID matching of three consecutive time frames in a confocal data sequence. (a), (b), and (c) are the three consecutive time frames from the time sequence data.

consecutive frames. Therefore, the quality of cell tracking really depends on the ID propagation process in Synthetic Brainbows. Although further improvements are possible to the measures in our method, we believe incorporating human interactions into the workflow is necessary for the most accurate result, as seen in our interactive segmentation techniques in Chapter 4. For massive time-dependent data, a more important task than tracking individual cells correctly is to visualize the general movement of different functional regions formed by these cells. In biology research, it is termed a fate map, which is a description of the origins of organs through different developmental stages. Since Synthetic Brainbows provide the movement of each cell, a time-dependent vector field can be then generated from our results. We believe that common vector field visualization techniques can be applied for studying fate maps.

CHAPTER 6

CONSTRUCTING ANATOMICAL ATLASES, A CASE STUDY

We worked with biologists and artists and developed a workflow for making anatomical atlases from confocal microscopy data. This workflow starts with acquiring confocal scans of mouse limbs, which are visualized using FluoRender's rendering pipeline (Chapter 3). Then, structures such as muscles, tendons, bones, and nerves are extracted using FluoRender's interactive segmentation functions (Chapter 4). The modeling process of the workflow first converts segmented volume data into coarse polygon models, and then these polygon models are processed with shrink-wrap simulations, which generate smooth and well-structured models. It is also fairly easy to unwrap the texture coordinates of these simulated models. Then, a digital painting package is used to transcribe textures from confocal scans to the polygon models. Finally, we use FluoRender for the presentation of the finished atlases, which are limbs of 14.5-day mouse embryos. This chapter is a detailed case study of how FluoRender can be used together with other software packages in practice.

6.1 Data Acquisition

The biologists working on the mouse limb atlas project are researching the cellular and molecular mechanisms governing the patterning and assembly of the musculoskeletal system during development. Understanding how the musculoskeletal system is assembled is a fundamental question in developmental biology. In addition, congenital defects in limb and musculoskeletal development are relatively common in humans, and understanding the etiology of these defects is of interest to the medical community. To study development of the limb

musculoskeletal system, the biologists have chosen to examine the development of the limb of mice. Mice are the primary model organism used to study limb musculoskeletal development. Not only are mouse and human development similar, but many genetic tools (the ability to create "knock-out" mice) and molecular reagents are also available in mice. To facilitate study of mouse limb development, the biologists wanted to create a 3D atlas of the developing mouse limb in which bones, tendons, muscles, and nerves were clearly displayed. Although an atlas of mouse limbs had previously been published by DeLaurier et al. [24], this atlas displayed a limited number of tissues (just muscles and bones) and details of muscle morphology (the orientation of the muscle fibers) were lacking.

Our workflow for constructing a limb atlas begins with obtaining digital images of the musculoskeletal system of mouse limbs. In a single mouse limb, tendons, muscles, and nerves were each labeled with a different fluorescently-tagged antibody. Limbs with fluorescently tagged tendons, muscles, and nerves were then imaged via confocal laser scanning microscopy. For each limb, a stack of in-register optical thin sections showing tendons in green, muscles in red, and nerves in blue were obtained. Bones were recognized as distinct black regions in the green and red channels. Figure 6.1 shows an acquired dataset of a mouse hind limb, which is visualized by volume rendering via FluoRender. Here, the three channels of muscles, tendons, and nerves are rendered in depth mode (Section 3.3, Chapter 3). The volume transfer function has been adjusted for each channel to remove noise and emphasize important structural information. The renderings are processed with tone mapping (Section 3.5, Chapter 3) and a shadow overlay (Section 3.6, Chapter 3) is used.

It should be noted that although this particular atlas was developed from data obtained via CLSM, our workflow and tools can also be applied to images obtained from other 3D imaging techniques, such as X-ray computed tomography (CT), magnetic resonance imaging (MRI), optical projection tomography (OPT), and high-resolution electron microscopy (HREM).

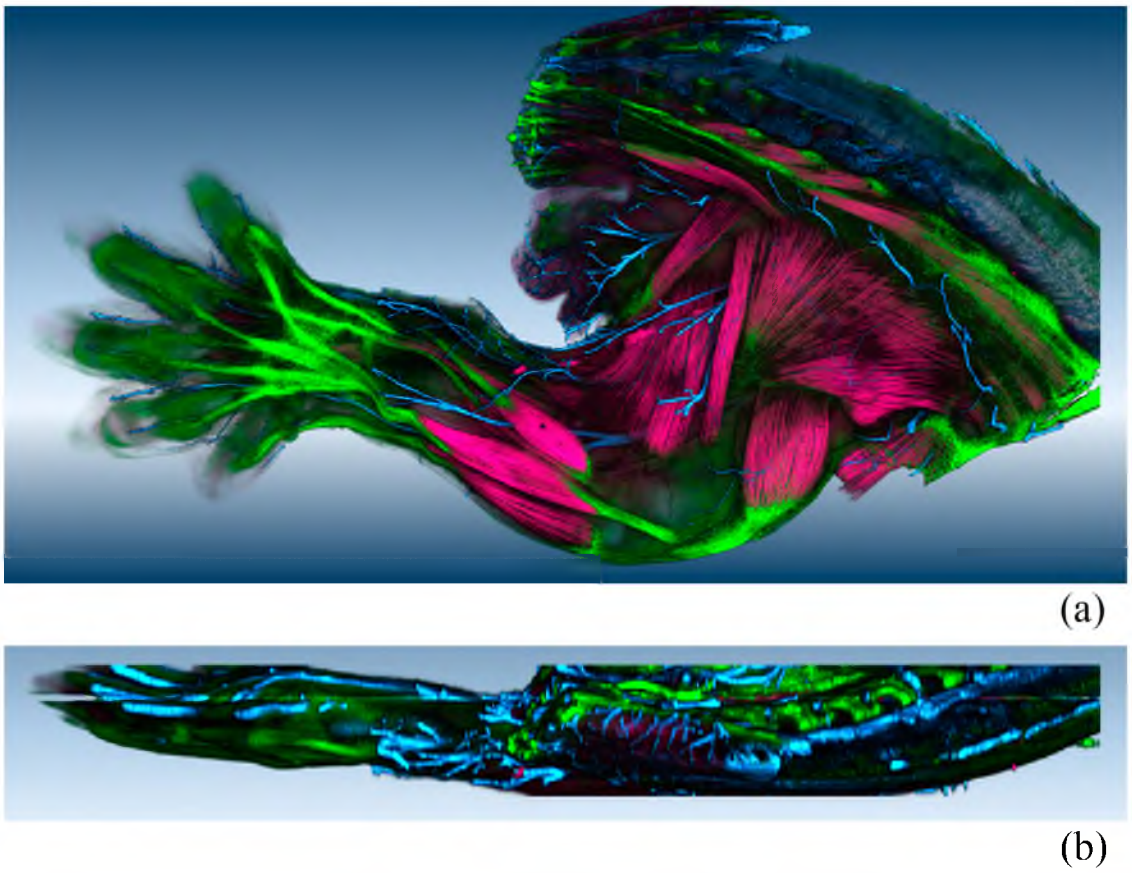

Figure 6.1. A volume-rendered visualization of the confocal data acquired for our atlas building project. This is the hind limb of a 14.5-day embryonic mouse. Muscles are shown in red, tendons in green, and nerves in blue. (a) When viewed in the XY plane, the visualization contains rich details of the structures. (b) When XZ plane is viewed, the visualization becomes coarse due to the increased Z increment.

6.2 Segmentation

For the mouse limb atlas, individual tendons, muscles, nerves, and bones need to be identified and segmented. The accuracy of segmenting these structures defines the quality of the final atlas. The real difficulty lies in the fact that the anatomy of mouse embryos is not entirely clear to researchers. We could use anatomies of adult rats and mouse as references, but there are always limitations and discrepancies, especially when each individual muscle needs to be identified and segmented. In FluoRender, we segment different types of structures with slightly different strategies. However, we always first try to segment them through

painting (Section 4.2, Chapter 4) on their visualized results whenever the structures are clear. Then, we restrict both the visualization and segmentation with clipping planes. Structures deep inside can usually be processed with this manner. For unclear and densely packed structures, we further restrict the clipping planes to a single slice, and use all the enhancements available (Section 3.6, Chapter 3). This is equivalent to segmenting slice-by-slice. However, FluoRender provides this special design of clipping planes that allows users to quickly go back and forth between view-dependent and slice-based segmentation (Section 4.3, Chapter 4).

• Muscles. They are the most important part of our limb atlas. They are also difficult to extract because they are densely packed and have obscure boundaries. Fine fibers imaged within each muscle further complicate the scenario. As in a common segmentation workflow in FluoRender, we first adjust the transfer function of a muscle channel. We usually decrease its gamma to enhance contrast, and cut off low-intensity noise by increasing threshold. Since large specimens like mouse limbs often have lower antibody penetration, thus lower signal intensity, we also decrease the saturation point to brighten the results. Tone-mapping operators are always used to further enhance the results. Depending on different regions of the visualization, for example, deep regions usually have lower signal intensities than surface regions. Parameters for transfer function and tone mapping are also adjusted during segmentation. Muscles in the foot area and major surface limb muscles are easily distinguished and then paint-selected, very possibly from various view directions. Selected muscles are progressively removed from the muscle channel through FluoRender's channel calculations (Section 4.3, Chapter 4). Then, deep muscles are exposed and more easily identified, which are selected and extracted next. This process resembles the actual dissection of an embryo under microscope (Figure 6.2). The segmented muscles are put together and compared with the original muscle channel. They are then correctly named according to their relative locations. Some difficult muscles are inspected and compared slice-by-slice. It is common that some muscles cannot be identified, in which case we can use other channels from the same scan to facilitate identification.

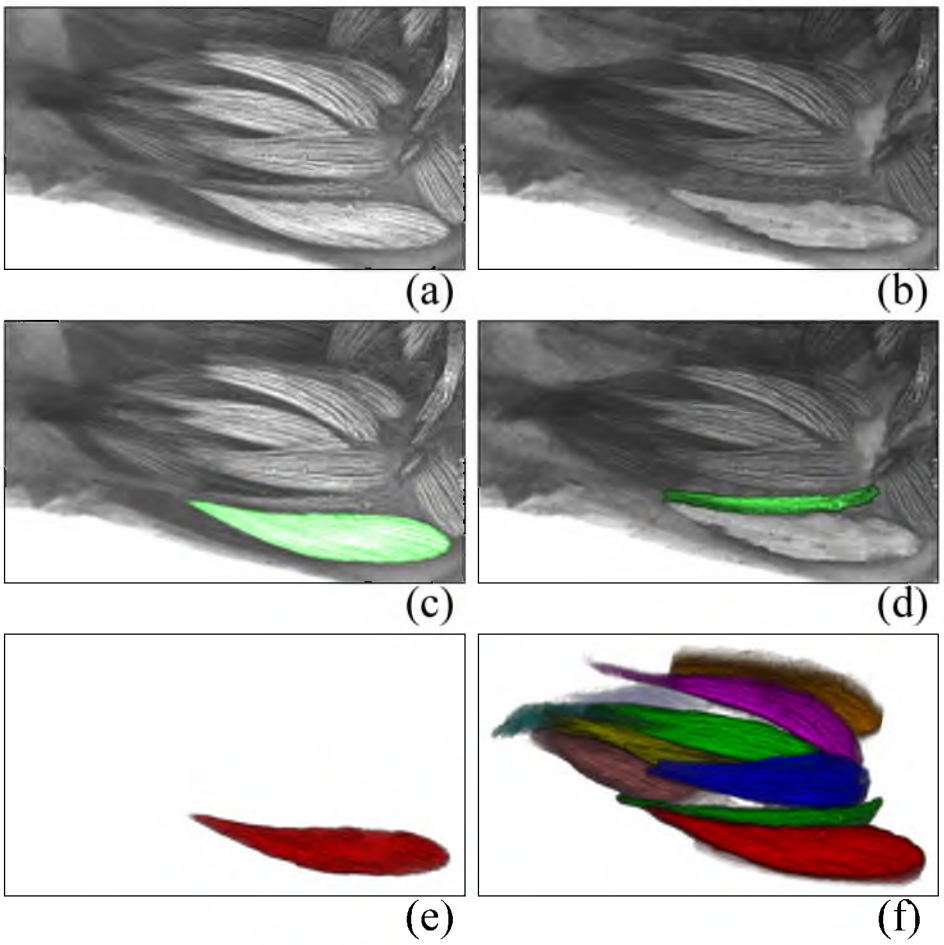

Figure 6.2. Segmentation of muscles from a mouse limb dataset. (a) The original muscle channel is visualized in FluoRender. (b) We paint directly on the visualization and select one muscle. (c) The selected muscle is extracted. (d) The selected muscle is removed from the original data, which makes selecting other muscles easier. (e) A second muscle is selected. (f) A group of muscles are segmented similarly.

For example, tendons of digitorum muscles are long and easily distinguishable; therefore, muscles connecting with these tendons are indisputably digitorum muscles.

• Tendons. Tendons attach muscles to bones. Most tendons in an embryo are blurry transitional area between bones and muscles. It is not practical to segment these tendons because of their indefinite shapes in confocal scans. We only extract long and distinguishable tendons such as tendons of digitorum muscles. In the

modeling section (Section 6.3), we will discuss how to model certain tendons without segmentation.

• Bones. Bones are not fluorescently stained in our experiments, and are black regions in the muscle and tendon channels. We segment them by inverting the intensity values of the muscle channel (or a combined channel from muscles and tendons, depending on the clarity of a specific scan) as in Figure 4.8. Thus, bones can be identified and extracted.

• Nerves. Similar in their shape to single neurons, nerves have extremely complex and branching structures in 3D. They can only be extracted by painting with FluoRender's selection brushes (Section 4.2, Chapter 4). For complex networks of nerves, such as the brachial plexus, different major branches are extracted separately. We use the selection brush to first select the starting point for a major nerve branch. Then, we use the diffusion brush to grow the selection of each major branch.

6.3 Modeling

The marching-cubes algorithm [64] is used to generate polygon models from the segmented structures. These automatically generated polygon models have very low qualities. Improvement of quality is possible through a variety of automatic algorithms. However, for the best visual quality, we developed a semi-automatic modeling workflow. This is due to following reasons. First, it is difficult for most automatic algorithms to generate a quad mesh, which is considered the basis for good structuring of a model. Second, the efficiency of polygon placement is low. The polygon contours often do not follow meaningful structures of the segmented volume. Instead, they usually congregate at noisy regions and form distractive patterns. Third, automatic algorithms often generate high polygon density, which makes further manual adjustment difficult, if not impossible. Fourth, there is no algorithm that can handle complicated shapes and generate high-quality models, such as those we see in the nervous system. Lastly, there is no guarantee to generate the same mesh topology for similar structures.

The same topology makes comparison between similar structures simple. For example, we can make a morphing animation very easily between to models if they have the same mesh topology.

We use Autodesk Maya [5] for making high-quality polygon models from marching-cubes results. Specifically, we use Maya's Nucleus simulation engine [7] to shrink-wrap a prototype model with a rubbery material onto the marching-cubes result. This method is semi-automatic, since the modeling and placement of the prototype model is manual, and shrink-wrapped models may also be manually adjusted for better shapes. In addition, branching structures, especially nerves, are not easily modeled by shrink-wrap. Manual modeling is needed for those structures.

Nonbranching structures including muscles and bones are modeled in the following workflow.

• Import. Marching-cubes generated polygon models are exported as individual OBJ files. These files can be easily imported into Maya. If FluoRender is used for both generating OBJ files and visualizing final models, unit and scale should match between final models and original confocal channels. If other formats or software are used, adjustment to unit and scale may be necessary for correct final presentations, especially when original confocal datasets are anisotropic.

• Build prototype models. A prototype model is a low detailed polygon model that completely encloses the marching-cubes generated model. Several qualities of the prototype model ensure successful shrink-wrap simulations, which then need little to no manual adjustments. First, there has to be sufficient space between a prototype model and an imported model. When the prototype model is shrunk onto the imported model, collision detection is calculated. Contact or intersection between them will generate strong collision force, which may drag the prototype model completely inside or make the simulation unstable. Second, the prototype model has to be composed of only quads. Quads can generate evenly distributed forces in physics-based simulations. The finished models are also easily subdivided to make high quality models. Most importantly, only a quad

mesh can have continuous contours that can be manipulated to follow important biological features, such as muscle fibers and ridges on bones. Third, the prototype model has to follow the shape of the imported model as close as possible and use as few vertices as possible. This usually requires the work of an experienced artist or trained modeler. However, similarities among different structures make it possible for sharing prototype models. For example, most muscles of limbs have a spindle shape. A spindle-shaped prototype muscle is prebuilt for all such muscles. Similarly, long bones of limbs can also share a tubular-shaped prototype model. Figure 6.3 demonstrates building prototype models for several typical shaped muscles. For irregular shaped structures, prototype models are built similarly, but they require more time for shape adjustments.

• Shrink-wrap. We create an nMesh passive collider from the imported model and an nCloth node from the prototype model. We set these parameters in the nCloth node's attribute editor: stretch resistance, bend resistance, rest length scale. All other dynamic properties are set to 0. We turn off gravity influence for the nCloth node. Then, we select the vertices of the prototype model and create a "point to surface" constraint onto the imported model. We set the type of this constraint to "'rubber bands"' and decrease the rest length of the constraint to a small value. The result of this setup is a mass-spring system, in which the edges of the prototype model will shrink and the vertices are dragged away from the normal directions of the imported model. We run this simulation and normally the result converges within seconds. To fixate the simulated model, we have to delete the construction history nodes and simulation nodes in Maya. An example of using shrink-wrap to model the extensor digitorum muscle is shown in Figure 6.4.

• Fine-tune. We then fine-tune models from shrink-wrap using Maya's standard polygon modeling tools. The amount of work in this step depends on the quality requirement of the final result. To generate fine detailed models, the result from shrink-wrap is converted to a subdivision model. Details can be added and manipulated at different detail levels of the subdivision model.

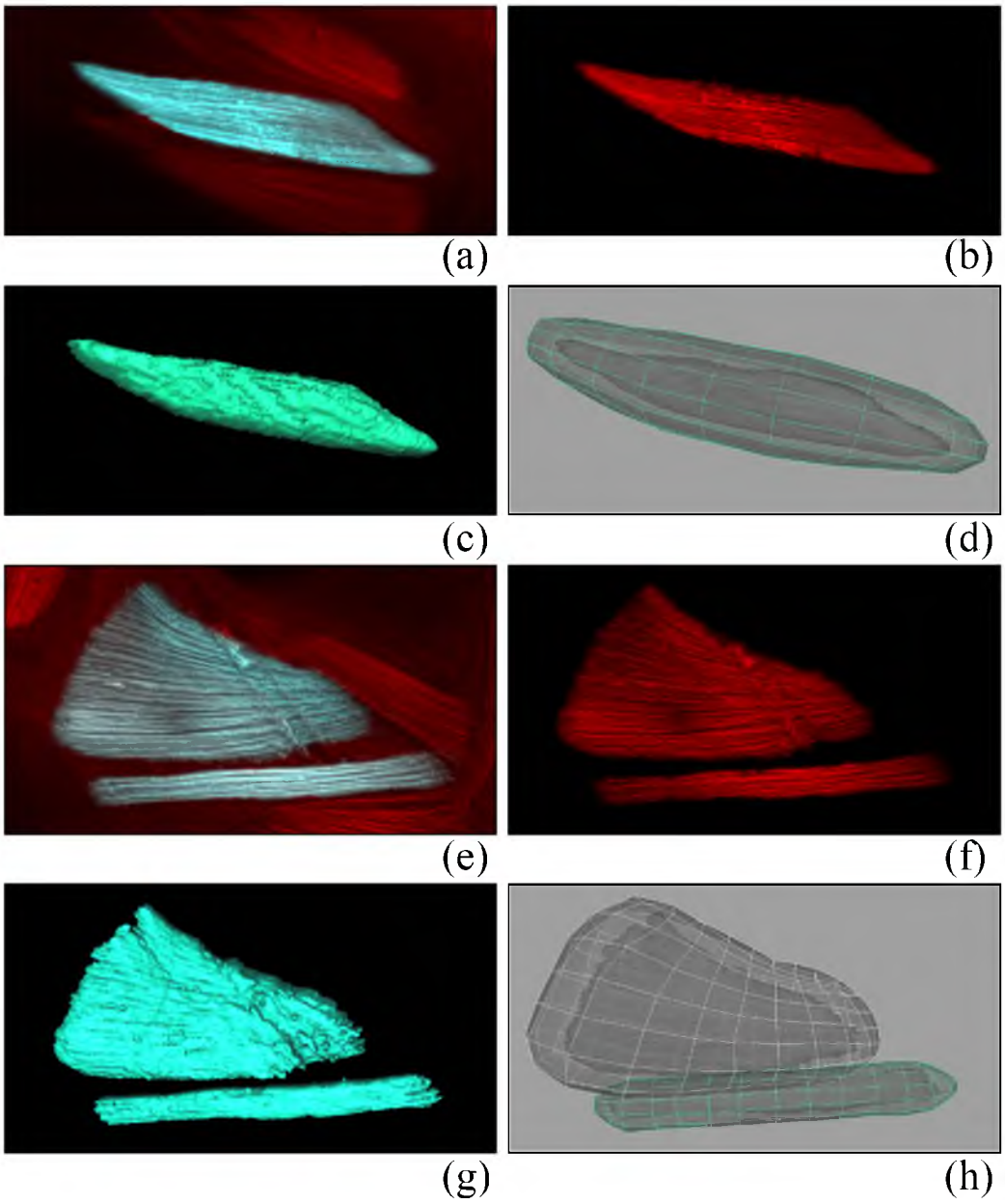

Figure 6.3. Building prototype models for different kinds of muscles. (a) The extensor digitorum longus muscle is selected in FluoRender. (b) The muscle is extracted. (c) In FluoRender, the muscle is converted to a polygon mesh with the marching-cubes algorithm. (d) In Maya, the polygon mesh generated in FluoRender is imported. A prototype model is built according to the imported model. This muscle has a very common shape for limb muscles, i.e., a spindle shape. (e) Two heads of the biceps femoris muscle are selected. (f) The two heads are extracted. (g) The two heads are converted to polygon models. (h) Prototype models are built according to these imported models.

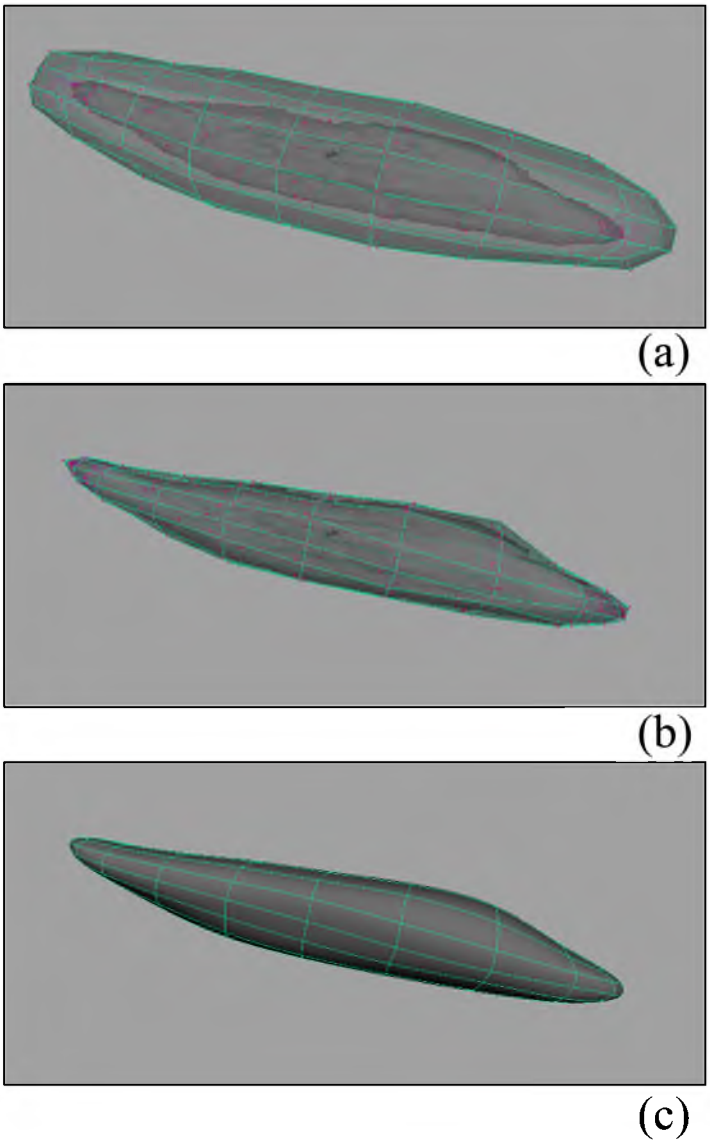

Figure 6.4. Modeling a muscle using shrink-wrap. (a) A marching-cubes-generated model is imported and a prototype model is built surround it. A passive collider is created from the marching-cubes-generated model. An nCloth object is created from the prototype model. Magenta connections represent the "point to surface" constraint between the two models. (b) We start the simulation. The prototype model is shrunk to the marching-cubes-generated model. (c) The final model is generated by smoothing the shrink-wrap simulation result. Notice that this setup lets polygon contours of the prototype model follow biological features, i.e., the muscle fiber directions.

As discussed in Section 6.2, some short tendons have obscure shapes, which are not segmented. They have to be modeled differently. We first finish the modeling of all the muscles, and then identify those muscles with tendons that are not segmented. Each muscle without segmented tendons is converted to an nMesh passive collider. Next, we build a cap-shaped model and then shrink-wrap it on one end of the muscle. The shrink-wrap process for these tendons is the same as that for the muscles. These tendon models are placed in the atlas to give visual presentations of the connections between muscles and bones. They can be further adjusted according to volume visualizations in FluoRneder.

Long and branching tendons may be modeled by shrink-wrap, considering they can be separated into several tubular-shaped parts. Nerves, however, are difficult to be modeled by shrink-wrap. Hence, we start from roots of nerves and extrude one face of a polygon cube. The extrusion is manually controlled to follow the branches of nerves. Since nerves usually do not present higher-level of complexity on each branch, the extrusion operation is adequate for completing all nerve models.

6.4 Texturing

Polygon models alone have good shape representations of the structures, but lack the details defining certain anatomical features, such as the muscle fibers. Commonly seen in the illustrated anatomy books, these features are not easily modeled with the methods discussed above. Sculpting, texture painting, or a combination of the two are used for adding realism to models in a CG artist's workflow. We choose texture painting for the anatomical details, mainly the muscle fibers, since textured polygon models are easily supported for final presentations.

Before textures can be applied, texture coordinates of the polygon model need to be created and mapped into a unit square of the texture space. This process is often referred as UV unwrapping. We perform this process to the prototype models only, since they have only a few vertices/UVs to manipulate

and their texture coordinates can be automatically interpolated when we smooth the models. Figure 6.5 is an example of unwrapping UVs of a typical muscle model. To unwrap UVs, we first use Maya's automatic UV projection (Figure 6.5 (a)). It generates separate UV pieces (Figure 6.5 (b)). We then use UV stitching to stitch these pieces together. The stitched UVs usually have entangled and highly distorted polygon faces (Figure 6.5 (c)). We then use Maya's automatic UV layout tool to obtain an improved layout (Figure 6.5 (d)). This layout usually has an adequate quality for further texture applications. However, depending on the methods that textures are applied, we may further edit this layout to give it a more regular structure (Figure 6.5 (e)).

• Shared textures. Models derived from the same prototype model can share one generic texture with no problem. Models from different prototypes but representing similar biological structures can share textures too. For example, muscle fibers are the main feature we want to add to muscle models. When building the muscle prototype models, a desired quality is that the contours of the quad mesh follow biological features (the muscle fibers). When we unwrap UVs for such particular prototype models, we know those contours represent muscle fiber directions. Thus, we lay out the UVs in a structure that the contours are aligned with one axis of the texture space. When a generic texture with vertical stripes is applied to a model with its UVs unwrapped this way, the stripes follow the contours and therefore the muscle fiber directions (Figure 6.6). When all prototype models of similar structures are laid out with this method, one generic texture with muscle fibers can be shared. This method is used to quickly make believable textures for visualizations of an atlas. One disadvantage of this method is that seams are usually visible, although using a well-designed repeating pattern and placing UVs carefully can reduce the artifacts. Another problem is that the two ends of these muscle models usually have highly distorted texture mappings, due to converging contours.

• Transcribed textures. Shared texture does not always meet the requirement of high precision and quality atlases. In addition, a generic texture can be highly

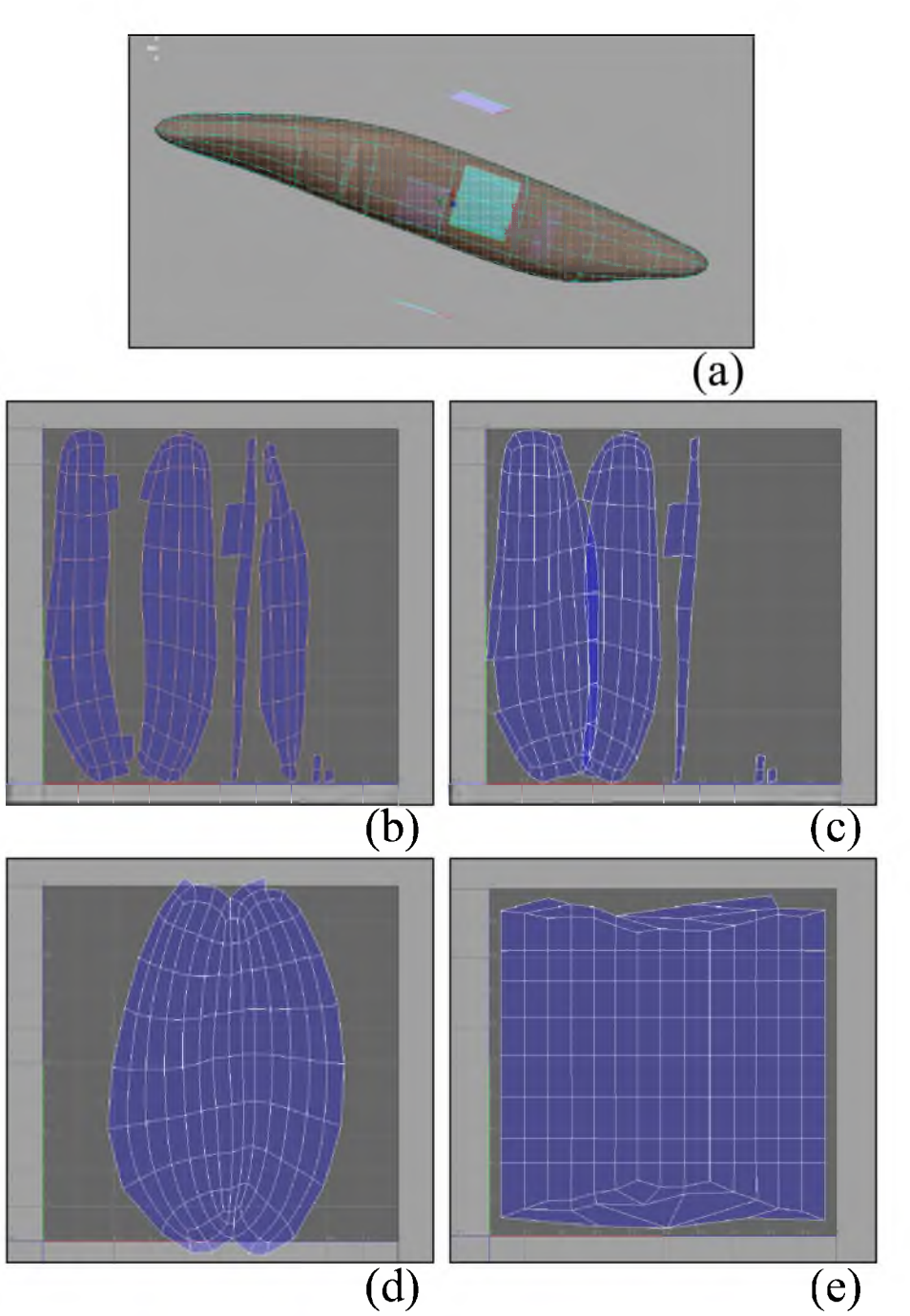

Figure 6.5. Unwrapping UVs of a muscle model. (a) Automatic UV projection is used to generate UVs. (b) In the texture coordinate space, the model is broken into several pieces. (c) Two pieces of the model's UVs are stitched together. (d) All UVs are stitched and laid out with Maya's automatic UV layout. (e) UVs are adjusted to be aligned with UV axes. This step is used so that generic muscle fiber texture can be shared among many models.

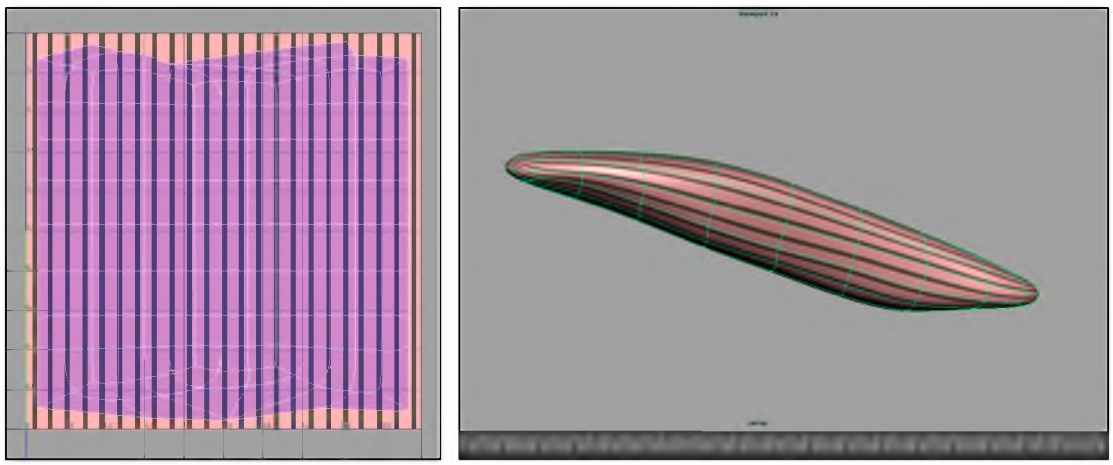

Figure 6.6. When UVs of a regularly-structured model are aligned with the texture space axes, a generic texture can be shared among models with similar structures. (a) The texture space, showing UVs of a muscle model and a texture with vertical stripes. (b) The textured model. The stripes of the texture are aligned with the model's contours, thus representing the directions of muscle fibers.

distorted on irregularly-shaped models. In order to get accurate presentations of some biological features such as the muscle fibers, we use a three-step procedure to transcribe muscle fibers from confocal visualization to polygon models (Figure 6.7). First, in FluoRender, we generate a volume rendering of a muscle. Next, we import this volume rendering into Photoshop [3]. This volume rendering is used as a reference layer. We paint on a semitransparent layer on top of the reference layer, and generate an image of illustrative fiber patterns, which should exactly follow the underlying muscle fiber directions. Finally, the illustrative pattern is imported into Autodesk Mudbox [6] along with the muscle model. The model is adjusted to the same view direction as in FluoRender. The illustrative pattern is used as a stencil. Mudbox's projection brush is used to project the stencil image onto the polygon model, similar to how paint brush works in FluoRender. This is too a user-guided process, so that we can progressively paint the stencil pattern on the model. In the meanwhile, both the stencil and the model can be manipulated so that areas initially at glancing angles can also be covered when the view is rotated. This is another place in our workflow where intensive manual work is

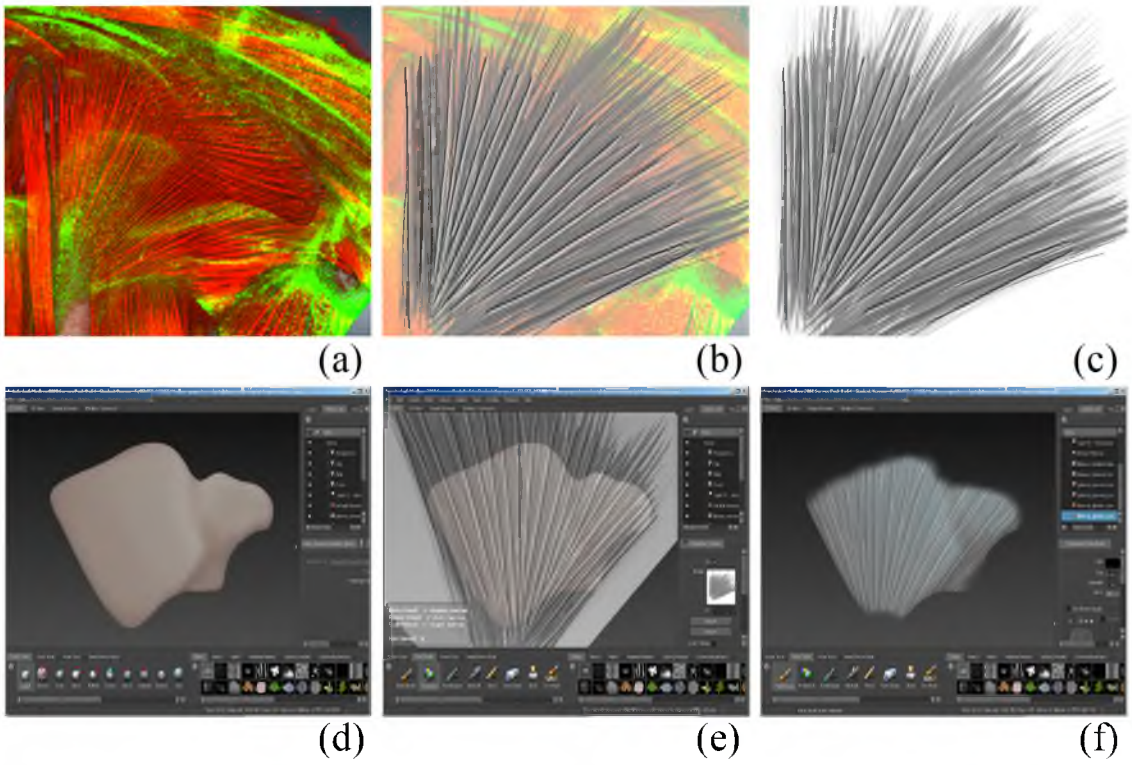

Figure 6.7. Texture transcription using Mudbox. (a) We load the rendering of the gluteus maximus muscle into Photoshop. (b) We generate patterns according to the volume-rendered result. (c) The image serves as a stencil. (d) We load the polygon model of the muscle into Mudbox. (e) After loading the stencil, we use Mudbox's projection brush to paint the stencil onto the model. (f) The stencil image is transcribed onto the model.

required, perhaps by an expert or well-trained user. The best illustrative quality can be achieved through this method. However, each model generated with this method has its individual texture, which increases the size of final atlases.

6.5 Results

We export finished atlases as individual model files in OBJ format, which can then be easily converted to many other polygon model formats if needed. The individual model files can be assembled and organized with a variety of model viewers. For interactively viewing the mouse limb atlas, we chose FluoRender, which is used to generate the final renderings in Figure 6.8. Because of its support of rendering semitransparent polygon models with depth peeling

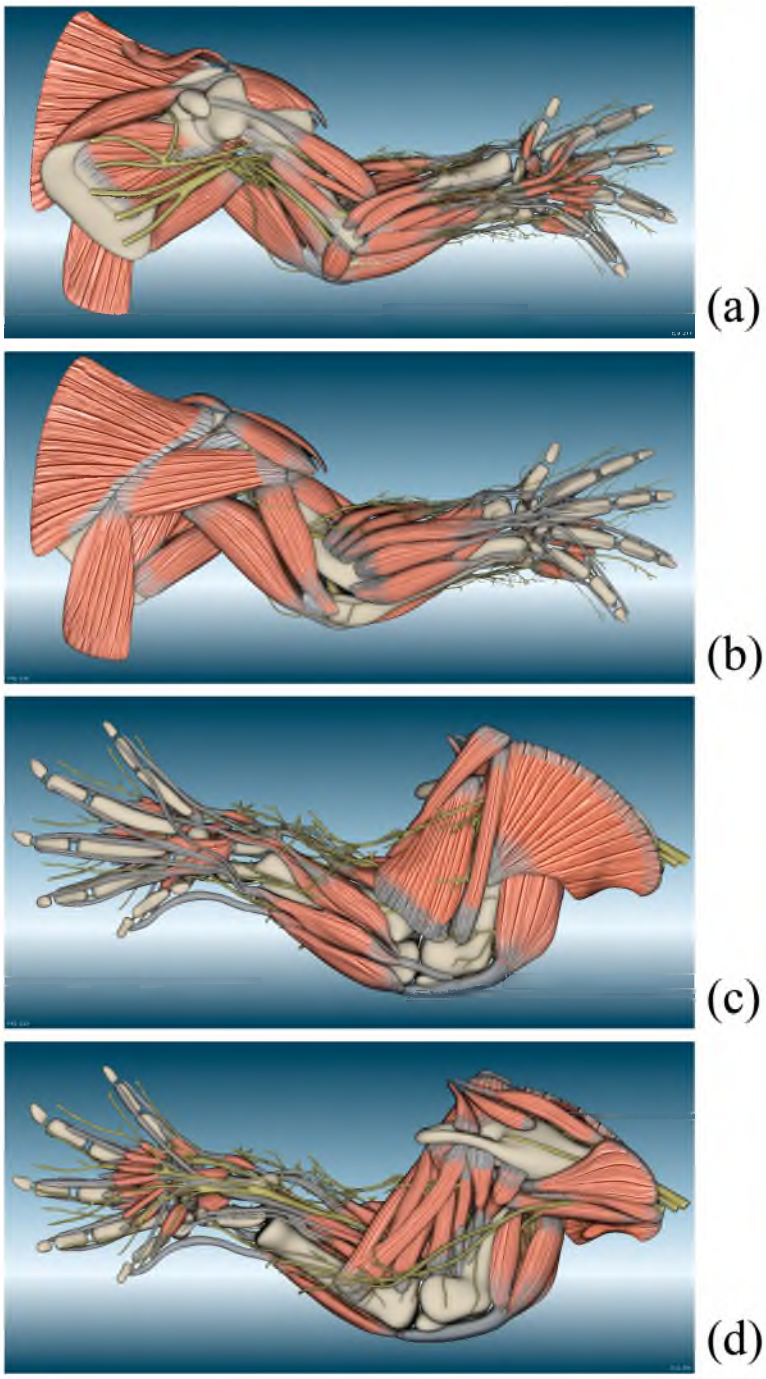

Figure 6.8. Limb atlases of 14.5-day mouse embryos. (a) The medial side of the forelimb. (b) The lateral side of the forelimb. (c) The lateral side of the hindlimb. (d) The medial side of the hindlimb.

(Section 3.4, Chapter 3), users can easily adjust the transparency and focus of structures while maintaining an informative context. As a volume rendering tool, FluoRender also enables us to simultaneously view the original volume data with the polygon-based atlas.

Figure 6.8 shows two limb atlases that we have built using the above workflow. A forelimb atlas (Figure 6.8 (a) and (b)) and a hindlimb atlas (Figure 6.8 (c) and (d)) of 14.5-day mouse embryos were built. These models along with FluoRender will be freely available to biologists researching limb muscles. Each muscle, tendon, bone, and nerve of these models has been annotated. Users are able to directly click on the models and obtain the information about the structures.

Segmentation with FluoRender and shrink-wrap modeling with Maya allow us to generate quality polygon models quickly from confocal scans. It not only enables us to build atlases of standard anatomy, but also makes comparisons between mutants and atlases easy. Mutants are genetically modified biological samples showing certain anomalies. Biologists want to compare and visualize the differences between mutants and standard anatomies. A similar workflow for making the limb atlas can then be used to generate polygon models of mutant mouse limb samples. There are several advantages of using this workflow for mutant model building and comparison. Firstly, researchers can create polygon models in an interactive and controllable manner. They can make adjustments to these models according to their interpretations of original data, which can usually yield high quality models. Secondly, models are already available from the standard atlases. Less adjustment to these models is necessary for the mutant models. Textures associated with the standard atlases can be reusable. Lastly and most importantly, this workflow ensures that models of mutants and standard atlases share the identical topology. We can then compare these models either visually by generating morphing animations or analytically by designing shape descriptors.

CHAPTER 7

CONCLUSIONS AND FUTURE WORK

FluoRender has been a freely available visualization and analysis tool for public download. We are happy to see it has been used in biological research in many laboratories across the world. Its visualization results can be found in several influential biology journals and image competitions. For biologists and medical researchers, visualization and analysis of volumetric data are in their developing stages. The progress FluoRender has made is due to a close collaboration between computer scientists and biologists. The success of such a development model requires us to find a common ground for the interests, knowledge, techniques, and requirements of collaborators. Still, there are more techniques in computer graphics and visualization that might be useful but have not been implemented; there are even more requirements from biologist users that could not be realized. Currently, FluoRender is a generalized tool for confocal data visualization and analysis. As we learned from our collaborating biologists, customized functions for many specialized biology research questions are demanded in practice. For example, after neural structures are extracted, their connections need to be mapped, and their length, width, and branches measured. Such measurements are usually different from structure to structure, or sample to sample. Customized functions and user interface building on top of a generalized tool can streamline researchers' workflow and shorten the time spent on visualization and analysis, which can be quite useful for repetitive experiments. However, integration of customized functions into a generalized tool does not seem to be a straightforward task. Also, there are very few successful examples that we can follow. For example, ImageJ [43] provides a scripting language

and lets customized modules call its existing functions. ImageJ became very popular among biologists because it can be customized and there exist many downloadable modules. Unfortunately, the lack of a strong volume visualization support in ImageJ makes streamlining workflows difficult. Furthermore, complex computations using scripts can be quite slow for volumetric data. A more appropriate approach should be developing customized systems based on strong visualization and intuitive analysis core functions. FluoRender is our first step towards this objective. However, re-engineering may be necessary in the future, if customized functions for specialized biology experiments are to be added easily.

The extension and customization of FluoRender can happen at different levels. Firstly, FluoRender provides settings for different sets of functions, such as volume rendering properties, 2D image space adjustment settings, and paint selection controls. Users may feel confused by looking at all the settings, or they may need different settings for different workflows. User can benefit from a "mode dial" system similar to that of a consumer-level digital camera: turning to a mode resets the settings and rearranges the user interface for a specific workflow. Secondly, extended functions should use existing user interactions as much as possible. For example, brushes are used for segmentation and proved to be intuitive. Many analyses can use this operation as well, such as finding the co-localization (co-localized structures stained by different fluorescent dyes) of multiple confocal channels. Finally, we learned that developing a tool for a biology research is just one part of designing a complete workflow. Similar to our existing functions and workflows presented here, we have to work with our collaborators to determine the proper workflow. Then, we customize or develop functions and make them into one tool that reflects this workflow.

APPENDIX

PUBLICATIONS

- "An Interactive Visualization Tool for Multi-channel Confocal Microscopy Data in Neurobiology Reseach",
  Yong Wan, Hideo Otsuna, Chi-Bin Chien and Charles Hansen,
  IEEE Transactions on Visualization and Computer Graphics, vol. 15, no. 6, 2009, pp. 1489-1496.

- "Fast Volumetric Data Exploration with Importance-Based Accumulated Transparency Modulation",
  Yong Wan and Charles Hansen,
  In *Proceedings of IEEE/EG International Symposium on Volume Graphics 2010*, pp. 61-68.

- "FluoRender: An Application of 2D Image Space Methods for 3D and 4D Confocal Microscopy Data Visualization in Neurobiology Research",
  Yong Wan, Hideo Otsuna, Chi-Bin Chien and Charles Hansen,
  In *Proceedings of IEEE Pacific Visualization Symposium (PacificVis)*, 2012, pp. 201-208.

- "A Practical Workflow for Making Anatomical Atlases in Biological Research",
  Yong Wan, A. Kelsey Lewis, Mary Colasanto, Mark van Langeveld, Gabrielle Kardon and Charles Hansen,
  IEEE Computer Graphics and Applications, vol. 32, no. 5, 2012, pp. 70-80.

- "Interactive Extraction of Neural Structures with User-Guided Morphological Diffusion",
  Yong Wan, Hideo Otsuna, Chi-Bin Chien and Charles Hansen,
  In *Proceedings of IEEE Symposium on Biological Data Visualization (BioVis)*, 2012, pp. 1-8.

- "Synthetic Brainbows",
  Yong Wan, Hideo Otsuna and Charles Hansen,
  Computer Graphics Forum, vol. 32, no. 3, 2013.

REFERENCES

[1] Abeysinghe, S., and Ju, T. Interactive skeletonization of intensity volumes. *The Visual Computer 25*, 5 (2009), 627–635.

[2] Adachi, S., and Lee, J. Computation by asynchronously updating cellular automata. *Journal of Statistical Physics 114*, 1-2 (2004), 261–289.

[3] Adobe. *Adobe Photoshop*, Mar. 2013. `http://www.photoshop.com/products/photoshop`.

[4] Akers, D. Cinch: a cooperatively designed marking interface for 3d pathway selection. In *Proceedings of the 19th annual ACM symposium on User interface software and technology* (2006), 33–42.

[5] Autodesk. *Autodesk Maya*, Mar. 2013. `http://usa.autodesk.com/maya/`.

[6] Autodesk. *Autodesk Mudbox*, Mar. 2013. `http://www.autodesk.com/mudbox`.

[7] Autodesk. *Nucleus in Autodesk Maya*, Mar. 2013. `http://images.autodesk.com/adsk/files/autodeskmaya_nucleus_whitepaper.pdf`.

[8] Baudet, G. M. Asynchronous iterative methods for multiprocessors. *Journal of the ACM 25*, 2 (1978), 226–244.

[9] Biafore, M. Cellular automata for nanometer-scale computation. *Physica D: Nonlinear Phenomena 70*, 4 (Feb. 1994), 415–433.

[10] Bitplane AG. *Imaris*, 2011. `http://www.bitplane.com/go/products/imaris`.

[11] Branda, C. S., and Dymecki, S. M. Talking about a revolution: The impact of site-specific recombinases on genetic analyses in mice. *Developmental Cell 6*, 1 (2004), 7–28.

[12] Bruckner, S., and Gröller, M. E. Instant volume visualization using maximum intensity difference accumulation. *Computer Graphics Forum 28*, 3 (2009), 775–782.

[13] Bruckner, S., Rautek, P., Viola, I., Roberts, M., Sousa, M. C., and Gröller, M. E. Hybrid visibility compositing and masking for illustrative rendering. *Computers and Graphics 34*, 4 (2010), 361–369.

[14] Bürger, K., Krüger, J., and Westermann, R. Direct volume editing. *IEEE Transactions on Visualization and Computer Graphics 14*, 6 (2008), 1388–1395.

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

[16] Cannon, J. R. *The One-Dimensional Heat Equation*, first ed. Addison-Wesley and Cambridge University Press, 1984.

[17] Chazan, D., and Miranker, W. Chaotic relaxation. *Linear Algebra and its Applications 2*, 2 (1969), 199–222.

[18] Chen, H.-L. J., Samavati, F. F., and Sousa, M. C. Gpu-based point radiation for interactive volume sculpting and segmentation. *The Visual Computer 24*, 7 (2008), 689–698.

[19] Claxton, N. S., Fellers, T. J., and Davidson, M. W. *Laser Scanning Confocal Microscopy*, 2008. `http://www.olympusconfocal.com/theory/confocalintro.html`.

[20] Cohen, A., Roysam, B., and Turner, J. Automated tracing and volume measurements of neurons from 3-d confocal fluorescence microscopy data. *Journal of Microscopy 173*, 2 (1994), 103–114.

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

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

[23] Correa, C., and Ma, K.-L. Visibility-driven transfer functions. In *PACIFICVIS '09: Proceedings of the 2009 IEEE Pacific Visualization Symposium* (Washington, DC, USA, 2009), IEEE Computer Society, 177–184.

[24] DeLaurier, A., Burton, N., Bennett, M., Baldock, R., Davidson, D., Mohun, T. J., , and Logan, M. P. The mouse limb anatomy atlas: An interactive 3d tool for studying embryonic limb patterning. *BMC Developmental Biology 8*, 83 (2008).

[25] di Stefano, L., and Bulgarelli, A. A simple and efficient connected components labeling algorithm. In *the 10th International Conference on Image Analysis and Processing* (1999), 322.

[26] Drebin, R. A., Carpenter, L., and Hanrahan, P. Volume rendering. In *SIGGRAPH '88: Proceedings of the 15th annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 1988), ACM, 65–74.

[27] Ebert, D., and Rheingans, P. Volume illustration: non-photorealistic rendering of volume models. In *Proceedings of the conference on Visualization '00* (2000), 195–202.

[28] Everitt, C. *Interactive Order-Independent Transparency*. White paper, Nvidia, 1999.

[29] Fellers, T. J., Vogt, K. M., and Davidson, M. W. *CCD Signal-To-Noise Ratio*, 2008. `http://www.microscopyu.com/tutorials/java/digitalimaging/signaltonoise/index.html`.

[30] Galilée, B., Mamalet, F., Renaudin, M., and Coulon, P.-Y. Parallel asynchronous watershed algorithm-architecture. *IEEE Transactions on Parallel and Distributed Systems 18*, 1 (2007), 44–56.

[31] Gao, Y., Yang, J., Xu, X., and Shi, F. Efficient cellular automaton segmentation supervised by pyramid on medical volumetric data and real time implementation with graphics processing unit. *Expert Systems with Applications 38*, 6 (2011), 6866–6871.

[32] Gilroy, A. M., MacPherson, B. R., and Ross, L. M. *Atlas of Anatomy, 1st ed.* Thieme Medical Publishers, 2009.

[33] Gonzalez, R. C., and Woods, R. E. *Digital Image Processing*, third ed. Prentice Hall, 2008.

[34] Gray, H., and Carter, H. V. *Anatomy: Descriptive and Surgical*. J.W. Parker, 1858.

[35] Greene, E. C. *Anatomy of the Rat*, reprint 1968 ed. Hafner Publishing Company, Inc., 1968.

[36] Grimm, S. *Real-Time Mono- and Multi-Volume Rendering of Large Medical Datasets on Standard PC Hardware*. PhD thesis, Vienna University of Technology, Gaullachergasse 33/35, 1160 Vienna, Austria, February 2005.

[37] Gustavson, S. *Simplex noise demystified*, Mar. 2005. `http://webstaff.itn.liu.se/~stegu/simplexnoise/simplexnoise.pdf`.

[38] Hauser, H., Mroz, L., Bischi, G. I., and Gröller, M. E. Two-level volume rendering. *IEEE Transactions on Visualization and Computer Graphics 7*, 3 (2001), 242–252.

[39] Hawick, K., Leist, A., and Playne, D. Parallel graph component labelling with gpus and cuda. *Parallel Computing 36*, 12 (2010), 655–678.

[40] Höhne, K. H., Bomans, M., Riemer, M., Schubert, R., Tiede, U., and Lierse, W. A volume-based anatomical atlas. *IEEE Computer Graphics and Applications 12* (1992), 72–78.

[41] Hosssain, Z., and Möller, T. Edge aware anisotropic diffusion for 3d scalar data. *IEEE Transactions on Visualization and Computer Graphics 16*, 6 (2010), 1376–1385.

[42] Huberman, B. A., and Glance, N. S. Evolutionary games and computer simulations. *PNAS 90*, 16 (1993), 7716–7718.

[43] ImageJ. *ImageJ, Image Processing and Analysis in Java*, Mar. 2013. `http://rsb.info.nih.gov/ij/index.html`.

[44] Jeong, W.-K., Beyer, J., Hadwiger, M., Vazquez, A., Pfister, H., and Whitaker, R. T. Scalable and interactive segmentation and visualization of neural processes in em datasets. *IEEE Transactions on Visualization and Computer Graphics 15*, 6 (2009), 1505–1514.

[45] Jobson, D., Rahman, Z., and Woodell, G. A multiscale retinex for bridging the gap between color images and the human observation of scenes. *IEEE Transactions on Image Processing 6*, 7 (1997), 965 –976.

[46] Ju, T. *Building a 3D Atlas of the Mouse Brain*. PhD thesis, Rice Univ., 2005.

[47] Kauffmann, C., and Piché, N. Cellular automaton for ultra-fast watershed transform on gpu. In *19th International Conference on Pattern Recognition* (2008), 1–4.

[48] Kauffmann, C., and Piché, N. Seeded nd medical image segmentation by cellular automaton on gpu. *International Journal of Computer Assisted Radiology and Surgery 5*, 3 (2010), 251–262.

[49] Kim, E., Shen, T., and Huang, X. A parallel cellular automata with label priors for interactive brain tumor segmentation. In *IEEE 23rd International Symposium on Computer-Based Medical Systems (CBMS)* (2010), 232–237.

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

[51] Kitware Inc. *Insight Toolkit*, 2011. `http://www.itk.org/`.

[52] Kitware Inc. *Visualization Toolkit*, 2011. `http://www.vtk.org/`.

[53] Kniss, J., Kindlmann, G., and Hansen, C. Multidimensional transfer functions for interactive volume rendering. *IEEE Transactions on Visualization and Computer Graphics 8*, 3 (2002), 270–285.

[54] Kniss, J., and Wang, G. Supervised manifold distance segmentation. *IEEE Transactions on Visualization and Computer Graphics 17*, 11 (2011), 1637–1649.

[55] Kreeger, K., and Kaufman, A. Mixing translucent polygons with volumes. *In Proceedings of IEEE Visualization 1999* (1999), 191–198.

[56] Kuhn, G., Oliveira, M., and Fernandes, L. An efficient naturalness-preserving image-recoloring method for dichromats. *IEEE Transactions on Visualization and Computer Graphics 14*, 6 (2008), 1747–1754.

[57] Kutulakos, K., and Seitz, S. A theory of shape by space carving. In *Computer Vision, 1999. The Proceedings of the Seventh IEEE International Conference on* (1999), vol. 1, 307–314.

[58] Lefohn, A. E., Kniss, J. M., Hansen, C. D., and Whitaker, R. T. Interactive deformation and visualization of level set surfaces using graphics hardware. In *Proceedings of the 14th IEEE Visualization 2003 (VIS'03)* (2003), 75–82.

[59] Levoy, M. S. *Display of surfaces from volume data*. PhD thesis, University of North Carolina at Chapel Hill, Chapel Hill, NC, USA, 1989.

[60] Li, G., Liu, T., Tarokh, A., Nie, J., Guo, L., Mara, A., Holley, S., and Wong, S. T. 3d cell nuclei segmentation based on gradient flow tracking. *BMC Cell Biology 8*, 40 (2007).

[61] Liu, J., Sun, J., and Shum, H.-Y. Paint selection. *ACM Transactions on Graphics 28*, 3 (2009), 69:1–69:7.

[62] Liu, Y., Cheng, H. D., Huang, J., Zhang, Y., and Tang, X. An effective approach of lesion segmentation within the breast ultrasound image based on the cellular automata principle. *Journal of Digital Imaging 25*, 5 (2012), 580–590.

[63] Livet, J., Weissman, T. A., Kang, H., Draft, R. W., Lu, J., Bennis, R. A., Sanes, J. R., , and Lichtman, J. W. Transgenic strategies for combinatorial expression of fluorescent proteins in the nervous system. *Nature 450* (Nov. 2007), 56–62.

[64] Lorensen, W. E., and Cline, H. E. Marching cubes: A high resolution 3d surface construction algorithm. *SIGGRAPH Computer Graphics 21*, 4 (1987).

[65] Luft, T., Colditz, C., and Deussen, O. Image enhancement by unsharp masking the depth buffer. *ACM Transactions on Graphics 25*, 3 (2006), 1206–1213.

[66] Machado, G. M., Oliveira, M. M., and Fernandes, L. A. F. A physiologically-based model for simulation of color vision deficiency. *IEEE Transactions on Visualization and Computer Graphics 15*, 6 (2009), 1291–1298.

[67] Margolus, N. Physics-like models of computation. *Physica D: Nonlinear Phenomena 10*, 1-2 (Jan. 1984), 81–95.

[68] Martin, W. N., and Aggarwal, J. K. Volumetric descriptions of objects from multiple views. *IEEE Transactions on Pattern Analysis and Machine Intelligence 5*, 2 (1983), 150–158.

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

[70] Nagy, Z., and Klein, R. Depth-peeling for texture-based volume rendering. *Proceedings of the 11th Pacific Conference on Computer Graphics and Applications* (2003), 429–433.

[71] Netter, F. H. *Atlas of Human Anatomy, 5th ed.* Elsevier, 2011.

[72] Nowak, M., and May, R. Evolutionary games and spatial chaos. *Nature 359* (1992), 826–829.

[73] Oliveira, V. M. A., and Lotufo, R. A. A study on connected components labeling algorithms using gpus. In *SIBGRAPI 2010* (2010).

[74] Olsen, Jr., D. R., and Harris, M. K. Edge-respecting brushes. In *Proceedings of the 21st annual ACM symposium on User interface software and technology* (2008), 171–180.

[75] Osher, S., and Sethian, J. A. Fronts propagating with curvature-dependent speed: algorithms based on hamilton-jacobi formulations. *Journal of Computational Physics 79*, 1 (1988), 12–49.

[76] Otsuna, H., and Ito, K. Systematic analysis of the visual projection neurons of drosophila melanogaster. i. lobula-specific pathways. *Journal of Comparative Neurology 497*, 6 (2006), 928–958.

[77] Owada, S., Nielsen, F., and Igarashi, T. Volume catcher. In *Proceedings of the 2005 symposium on Interactive 3D graphics and games* (2005), 111–116.

[78] Owada, S., Nielsen, F., Igarashi, T., Haraguchi, R., and Nakazawa, K. Projection plane processing for sketch-based volume segmentation. In *Proceedings of the 5th IEEE International Symposium on Biomedical Imaging: From Nano to Macro* (may 2008), 117–120.

[79] PerkinElmer Inc. *Volocity 3D Image Analysis Software*, 2011. `http://www.perkinelmer.com/pages/020/cellularimaging/products/volocity.xhtml`.

[80] Perona, P., and Malik, J. Scale-space and edge detection using anisotropic diffusion. *IEEE Transactions on Pattern Analysis and Machine Intelligence 12*, 7 (1990), 629–639.

[81] Reinhard, E., Ward, G., Pattanaik, S., and Debevec, P. *High Dynamic Range Imaging: Acuisition, Display and Image-Based Lighting*, first ed. Elsevier Inc., 2006.

[82] Rezk Salama, C., Keller, M., and Kohlmann, P. High-level user interfaces for transfer function design with semantics. *IEEE Transactions on Visualization and Computer Graphics 12*, 5 (2006), 1021–1028.

[83] Richardson, R. *The making of Mr. Gray's Anatomy, 1st ed.* Oxford Univ. Press, 2008.

[84] Rivest, J.-F., Soille, P., and Beucher, S. Morphological gradients. *Journal of Electronic Imaging 2*, 4 (1993), 326–341.

[85] Rossler, F., Tejada, E., Fangmeier, T., Ertl, T., and Knauff, M. Gpu-based multi-volume rendering for the visualization of functional brain images. *In Proceedings of SimVis 2006* (2006), 305–318.

[86] Rothman, D. H., and Keller, J. M. Immiscible cellular-automaton fluids. *Journal of Statistical Physics 52*, 3-4 (1988), 1119–1127.

[87] Saad, A., Hamarneh, G., and Möller, T. Exploration and visualization of segmentation uncertainty using shape and appearance prior information. *IEEE Transactions on Visualization and Computer Graphics 16*, 6 (2010), 1366 –1375.

[88] Saad, A., Möller, T., and Hamarneh, G. Probexplorer: Uncertainty-guided exploration and editing of probabilistic medical image segmentation. *Computer Graphics Forum 29*, 3 (2010), 1113–1122.

[89] Sato, T., Hamaoka, T., Aizawa, H., Hosoya, T., and Okamoto, H. Genetic single-cell mosaic analysis implicates ephrinb2 reverse signaling in projections from the posterior tectum to the hindbrain in zebrafish. *Journal of Neuroscience 27*, 20 (2007), 5271–5279.

[90] Sato, Y., Nakajima, S., Shiraga, N., Atsumi, H., Yoshida, S., Koller, T., Gerig, G., and Kikinis, R. Three-dimensional multi-scale line filter for segmentation and visualization of curvilinear structures in medical images. *Medical Image Analysis 2*, 2 (1998), 143–168.

[91] Segal, M., and Akeley, K. *The OpenGL Graphics System: A Specification*, Nov. 2012. `http://www.opengl.org/registry/`.

[92] Segall, C., and Acton, S. Morphological anisotropic diffusion. In *Proceedings of International Conference on Image Processing 1997* (Oct 1997), vol. 3, 348–351.

[93] Sherbondy, A., Houston, M., and Napel, S. Fast volume segmentation with simultaneous visualization using programmable graphics hardware. In *Proceedings of the 14th IEEE Visualization 2003 (VIS'03)* (2003), 171–176.

[94] Smith, M. A., Bar-Yam, Y., Rabin, Y., Margolus, N., Toffoli, T., and Bennett, C. H. Cellular automaton simulation of polymers. In *MRS Fall Meeting* (1991), vol. 248, 483–488.

[95] Soille, P. *Morphological Image Analysis: Principles and Applications*, second ed. Springer-Verlag, 2002.

[96] Sowell, R., Liu, L., Ju, T., Grimm, C., Abraham, C., Gokhroo, G., and Low, D. Volume viewer: an interactive tool for fitting surfaces to volume data. In *Proceedings of the 6th Eurographics Symposium on Sketch-Based Interfaces and Modeling* (2009), 141–148.

[97] Tay, T. L., Ronneberger, O., Ryu, S., Nitschke, R., and Driever, W. Comprehensive catecholaminergic projectome analysis reveals single-neuron integration of zebrafish ascending and descending dopaminergic systems. *Nature Communications 2* (2011), 171.

[98] Tikhonova, A., Correa, C., and Ma, K.-L. Explorable images for visualizing volume data. In *IEEE Pacific Visualization Symposium (PacificVis)* (2010), 177–184.

[99] Tikhonova, A., Correa, C., and Ma, K.-L. Visualization by proxy: A novel framework for deferred interaction with volume data. *IEEE Transactions on Visualization and Computer Graphics 16*, 6 (2010), 1551–1559.

[100] Tzeng, F.-Y., Lum, E. B., and Ma, K.-L. An intelligent system approach to higher-dimensional classification of volume data. *IEEE Transactions on Visualization and Computer Graphics 11*, 3 (2005), 273–284.

[101] Udupa, J. K., and Samarasekera, S. Fuzzy connectedness and object definition: Theory, algorithms, and applications in image segmentation. *Graphical Models and Image Processing 58*, 3 (1996), 246–261.

[102] Venkatasubramanian, S., and Vuduc, R. W. Tuned and wildly asynchronous stencil kernels for hybrid cpu/gpu systems. In *the 23rd International Conference on Supercomputing* (2009), 244–255.

[103] Vezhnevets, V., and Konouchine, V. "growcut" - interactive multi-label n-d image segmentation by cellular automata. In *Graphicon* (2005), 150–156.

[104] Vincent, L., and Soille, P. Watersheds in digital spaces: an efficient algorithm based on immersion simulations. *IEEE Transactions on Pattern Analysis and Machine Intelligence 13*, 6 (1991), 583–598.

[105] Viola, I., Kanitsar, A., and Gröller, M. E. Hardware-based nonlinear filtering and segmentation using high-level shading languages. In *Proceedings of the 14th IEEE Visualization 2003 (VIS'03)* (2003), 309–316.

[106] Visage Imaging. *Amira*, 2011. `http://www.amiravis.com`.

[107] von Neumann, J. *Theory of self-reproducing automata*. University of Illinois Press, 1966.

[108] Wallis, J., Miller, T., Lerner, C., and Kleerup, E. Three-dimensional display in nuclear medicine. *IEEE Trans. Medical Imaging 8*, 4 (1989), 297–303.

[109] Wallis, J., Miller, T., Lerner, C., and Kleerup, E. Three-dimensional display in nuclear medicine. *IEEE Transactions on Medical Imaging 8*, 4 (1989), 297–230.

[110] Wang, L., Giesen, J., McDonnell, K., Zolliker, P., and Mueller, K. Color design for illustrative visualization. *IEEE Transactions on Visualization and Computer Graphics 14*, 6 (2008), 1739–1754.

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

[112] West, D. B. *Introduction to Graph Theory*, 2nd ed. Prentice Hall, 1999.

[113] Yuan, X., Zhang, N., Nguyen, M. X., and Chen, B. Volume cutout. *The Visual Computer 21*, 8 (2005), 745–754.