EXPLORATION, REGISTRATION, AND ANALYSIS OF HIGH-THROUGHPUT 3D MICROSCOPY DATA FROM THE KNIFE-EDGE SCANNING MICROSCOPE

A Dissertation

by

CHUL SUNG

Submitted to the Office of Graduate and Professional Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

Chair of Committee,	Yoonsuck Choe
Committee Members,	John Keyser
	Jinxiang Chai
	Louise C. Abbott
Head of Department,	Nancy Amato

May 2014

Major Subject: Computer Science and Engineering

ABSTRACT

Advances in high-throughput, high-volume microscopy techniques have enabled the acquisition of extremely detailed anatomical structures on human or animal organs. The Knife-Edge Scanning Microscope (KESM) is one of the first instruments to produce sub-micrometer resolution ($\sim 1\mu m^3$) data from whole small animal brains. We successfully imaged, using the KESM, entire mouse brains stained with Golgi (neuronal morphology), India ink (vascular network), and Nissl (soma distribution). Our data sets fill the gap of most existing data sets which have only partial organ coverage or have orders of magnitude lower resolution. However, even though we have such unprecedented data sets, we still do not have a suitable informatics platform to visualize and quantitatively analyze the data sets.

This dissertation is designed to address three key gaps: (1) due to the large volume (several tera voxels) and the multiscale nature, visualization alone is a huge challenge, let alone quantitative connectivity analysis; (2) the size of the uncompressed KESM data exceeds a few terabytes and to compare and combine with other data sets from different imaging modalities, the KESM data must be registered to a standard coordinate space; and (3) quantitative analysis that seeks to count every neuron in our massive, growing, and sparsely labeled data is a serious challenge.

The goals of my dissertation are as follows: (1) develop an online neuroinformatics framework for efficient visualization and analysis of the multiscale KESM data sets, (2) develop a robust landmark-based 3D registration method for mapping the KESM Nissl-stained entire mouse data into the Waxholm Space (a canonical coordinate system for the mouse brain), and (3) develop a scalable, incremental learning algorithm for cell detection in high-resolution KESM Nissl data.

For the web-based neuroinformatics framework, I prepared multi-scale data sets at different zoom levels from the original data sets. And then I extended Google Maps API to develop atlas features such as scale bars, panel browsing, and transparent overlay for 3D rendering. Next, I adapted the OpenLayers API, which is a free mapping and layering API supporting similar functionality as the Google Maps API. Furthermore, I prepared multi-scale data sets in vector-graphics to improve page loading time by reducing the file size. To better appreciate the full 3D morphology of the objects embedded in the data volumes, I developed a WebGL-based approach that complements the web-based framework for interactive viewing. For the registration work, I adapted and customized a stable 2D rigid deformation method to map our data sets to the Waxholm Space. For the analysis of neuronal distribution, I designed and implemented a scalable, effective quantitative analysis method using supervised learning. I utilized Principal Components Analysis (PCA) in a supervised manner and implemented the algorithm using MapReduce parallelization.

I expect my frameworks to enable effective exploration and analysis of our KESM data sets. In addition, I expect my approaches to be broadly applicable to the analysis of other high-throughput medical imaging data.

DEDICATION

To my beloved mother and father,

Kyunghee Ka and Ukgi Sung, and my brother Min

ACKNOWLEDGEMENTS

First and foremost, I would like to express my sincere gratitude to my advisor, Prof. Yoonsuck Choe, for guiding me. During my Ph.D. training, he carefully observed my characteristics and my research, giving me precise guidance and suggesting better ways not only to do my research, but also to improve as a person. When I met obstacles in my research, his valuable comments made the challenges to be overcome easier.

I would also like to express my sincere gratitude to Prof. John Keyser and Prof. Louise C. Abbott. I was honored to meet them every week in the Brain Networks Laboratory (BNL) meetings. I had the opportunity to listen to their guidance and learn about working as an academic. I would also like to thank Prof. Jinxiang Chai for his time and comments, widening the point of view of my research.

In addition, I would like to thank the 3Scan company folks: Todd Huffman (CEO), Katy Pelton, Matthew Goodman, Megan Klimen, Cody Daniel, and Sean Kolk, for their enthusiasm, collaboration, and financial support. I am also grateful for the Korea Institute for the Advancement of Technology (KIAT) and SystemBase company (CEO, Myunghyun Kim) for their financial support.

Part of this research was funded by the National Science Foundation, under grants #0905041 and #1208174 (Collaborative Research in Computational Neuroscience, PI: Yoonsuck Choe). I am also grateful to Korean Scientists and Engineers Association, Korean Computer Scientists and Engineers Association, and Hyundai Motor for providing generous scholarships.

I have to thank my research collaborators: Prof. Jongwook Woo, Dr. Jiryang Chung, Daniel Miller, Dr. David Mayerich, Dr. Jaerock Kwon, Dr. DongHyeop Han,

Dr. Huei-Fang Yang, Jinho Choi, Jinsoo Kim, Mitchell Priour, Rachel Dorn, Han Wang, and Manisha Srivastava. Without their help and comments, this research would not have been possible. Particularly, Prof. Jongwook Woo gave me insight and background knowledge related to big-data analysis.

I would also like to thank my laboratory members and friends for their encouragement and support: Dr. Timothy Mann, Dr. Henry Choi, Jaewook Yoo, Noah Larsen, Junseok Lee, Zhuo Yue, Wencong Zhang, Wenjie Yang, Shashwat Lal Das, Pastor Sungsoo Kim, Dr. Donghun Kang, Pastor John Schmid, Robert Ulbrich, Jason Werner, Darlys Werner, Rachel Dawson, Deanna Grandalen, Jin Huang, Zhaoyan Xu, Wookyung An, Sungjun Lim, and Michael Nowak.

Finally, I am deeply grateful to my parents, Ukgi Sung and Kyunghee Ka and my brother and sister-in-law, Min Sung and Hyungkyung Yoon, for their constant support and prayers. Through all my research, I would like to attribute glory to God.

TABLE OF CONTENTS

LIST OF FIGURES

FIGURE Page

1. INTRODUCTION

The connectome is a complete collection of the neural connections of a whole brain [87, 86, 85]. Obtaining the connectomes of individual brains and their structural analysis provide critical information for understanding brain structures and their functions as well as the structural causes of brain dysfunction such as schizophrenia and autism. While the human connectome has been available for analysis at a centimeter resolution using non-invasive tools, high-throughput, high-volume microscopy techniques for small animal brains have enabled the description of connectivity at the scale of individual neurons and synapses [21].

The Knife-Edge Scanning Microscope (KESM) is one of the first instruments to produce sub-micrometer resolution ($\sim 1 \mu m^3$) data from whole small animal brains [55]. We successfully imaged, using the KESM, entire mouse brains stained with Golgi (neuronal morphology) [16], India ink (vascular network) [57], and Nissl (soma distribution) [14] at 0.6 μm $\times$ 0.7 μm $\times$ 1.0 μm resolution. These KESM data sets fill the gap of most existing data sets between having only partial organ coverage and having orders of magnitude lower resolution [55, 62, 59, 60, 61].

It is an important task to develop frameworks for disseminating and analyzing such full-scale neural and vascular microstructures of the mouse brain. My dissertation aims to tackle the challenges of developing these frameworks.

1.1 Motivation

KESM data sets can contribute greatly to neuroscience research because they provide high-resolution neuroanatomical structures of whole small animal brains. However, their dissemination, registration, and quantitative analysis are primary challenges.

KESM high-volume image data with sub-micrometer resolution can easily exceed a few terabytes. We can use online cloud systems to upload and share the KESM data sets, but users need to have their own high-capacity hard disk to download and save them. Moreover, it is physically and computationally expensive to render the volume data in 3D. Rendering large volumes in 3D becomes difficult without high-end graphics cards and extensive graphics memory. These challenges hinder public sharing of these large, high resolution data sets, thus posing major obstacles for collaboration amongst neuroscience researchers.

The KESM data sets are unique in their detail and extent compared to other currently available data sets. Submicrometer voxel size across entire small animal brains from KESM (or similar approaches) can fill the critical gap between large-scale, lower resolution methods like diffusion MRI [8, 27, 76, 94] on the one hand and small-scale, higher resolution methods like SBF-SEM [18] on the other hand. So, mapping our KESM data sets to a standard coordinate system for the mouse brain provides deeper understanding of the mouse brain by across referencing different mouse brain atlases. Also, due to the limitation of our neuroanatomy data sets without the neural activity information, mapping the KESM data sets to the functional atlas of the rat brain will help to compare anatomical structures with neural activities in their regions.

Analysis of neuronal distributions in the brain plays an important role in the understanding to organization and in the diagnosis of disorders of the brain. The KESM Nissl data set enables detailed studies of cortical and subcortical distribution of neuronal cell bodies. However, a quantitative analysis that seeks to count every neuron in our high-resolution Nissl data set is faced with a serious challenge.

1.2 Approach

There are several mouse brain atlases available, with data from different imaging modalities, but their resolution is not high enough in one or more of the x, y, or z axis to show the morphological detail of neurons or microvasculatures [90, 65, 98, 53].

To disseminate our high-resolution whole brain KESM data sets, I have developed a light-weight web-based 3D rendering approach, only using standard HTML, Javascript, and Cascading Styling Sheets (CSS) to achieve a quick, effective, and resource-efficient web-based interface.

The first version of the web-based KESM brain atlas (or KESMBA v1) extended the Google Maps API (`https://developers.google.com/maps/`) to include brain atlas features such as scale bars and panel browsing.

A challenging issue arises due to the 3D nature of the KESM data, in contrast to the 2D maps in Google Maps. In Google Maps, 2D image sets for different zoom levels are stored at the server side, and those are requested on-demand by the client. However, the client cannot just display 2D images from the KESM image stack. They need to view it in 3D somehow but that can cause a huge memory and computational overhead, regardless of whether it is done at the client side or the server side. Also, single 2D images may not be informative.

A good compromise would be to use transparency (alpha channel) in the stored images, so that clients can request multiple images from a stack and display it as an offset overlay to give a 3D effect. KESMBA v1 consists of tile images with improved contrast and added alpha channel. The client can control how many images to stack up. These overlays can be done with straight-forward HTML, Javascript, and Cascading Styling Sheets (CSS).

Efficient navigation requires multiple resolutions of the data and to reduce com-

munication bandwidth, data sets at different zoom levels need to be pre-generated and stored in our data server. KESMBA v1 supports not only original zoom level tiles, but the tiles of down-scaled data sets.

Although Google Maps API provides many functionalities for manipulating brain atlases, the API is licensed commercially. Because of that, we cannot locally store the API library and need to be connected the networks for using the API library. To solve these issues, I adapted the OpenLayers API (`http://openlayers.org/`) instead of Google Maps API for its mapping features as the second version of KESM brain atlas (or KESMBA v2). OpenLayers API is comparable to Google Maps API, but it is open source, a JavaScript library available under the GNU GPL license. Due to this, we can maintain a local, customized copy of the whole API library in our server.

Moreover, the addition of alpha channel increased the file size of individual tiles. Besides using the open source library, I was able to greatly reduce the file size of each tile by converting the original data to Scalable Vector Graphics (SVG) data, which is an XML-based vector image format, while retaining the complete structural information. Besides, the SVG format does not need to fill in its background, so I do not require an additional alpha channel to the individual tiles.

Furthermore, KESMBA v1 only supports the native orientation of the data sets: the Golgi data sets were obtained in horizontal sections; and the India ink and Nissl data sets were in coronal sections. Restricting to a single orientation hampers exploration and understanding of the data. To minimize bringing up unit volumes for fully interactive 3D inspection, a good alternative is to provide all three standard orientations: horizontal, coronal, and sagittal. KESMBA v2 has a pull-down menu to choose between these three standard orientations.

KESMBA v1 and v2 are ideal for surveying large volumes of data. However, since

the viewpoint is fixed, it can be hard to appreciate the full 3D morphology of the objects embedded in the data volumes. In order to overcome this, I have developed a WebGL-based approach that complements the KESM brain atlas for interactive viewing. Due to limits in computational power and memory of a typical personal computer, only a small volume of data can be loaded in a web browser and inspected at a time.

Besides the dissemination of KESM data sets, to enhance interoperability between the KESM mouse brain atlas and other mouse brain atlases, I have transformed our KESM Nissl data set into the Waxholm space (WHS), a new standard coordinate space for rodents [30]. The benefit of this step is mutual. We can pull detailed annotations at no cost from existing resources which include target volumes and over fifty 3D data sets in the WHS [30]. In turn, our high-resolution KESM data can provide rich anatomical data to other atlases.

As a WHS registration workflow, the KESM has unique noise characteristics, so my first task was to suppress noise in the KESM data set, adapting a Fast Fourier Transform (FFT)-based denoising algorithm. Then, in order to map the cleaned Nissl data set to WHS, I have developed a landmark-based 3D registration framework, extended a stable 2D rigid deformation method, for fine registration. This framework produces a high resolution mouse brain atlas that leverages on a large set of preexisting annotations in the WHS.

In addition to the WHS, the Allen Brain Atlas (ABA) website provides Allen Reference Atlases (ARAs) in the coronal and the sagittal plane to complement a genome-wide map of gene expression in the mouse brain. The reference atlases are a web-based systemically organized taxonomy of mouse brain structures and they also allow users to download full-color references of mouse brain structures in SVG format, which do not contain the titles of each structure.

I have developed a crawling tool to extract titled full-color references of mouse brain structures in SVG format from the ARAs. Once I have 132 SVG references and 21 SVG references in the coronal and sagittal plane, I manually mapped the Allen SVG references to our KESM mouse brain atlases and overlayed the Allen SVG references on KESMBA v2. This overlay plays a crucial role in annotations of our high-resolution moue brain data.

Accurate estimation of neuronal count and distribution is central to the understanding of the organization and layout of cortical maps and subcortical nuclei in the brain, and changes in the cell population induced by brain disorders. To conduct such a quantitative analysis of the neuron distribution, I have developed a scalable, incremental learning algorithm for cell body detection. This algorithm is computationally efficient (linear mapping, non-iterative) and does not require retraining (unlike gradient-based approaches) or retention of old raw data (unlike instance-based learning).

I have tested my algorithm on our rat brain Nissl data set, showing superior performance compared to an artificial neural network-based benchmark. I have also demonstrated robust performance in a scenario where the data set is rapidly growing in size. The algorithm is also highly parallelizable due to its incremental nature, and I have demonstrated this empirically using a MapReduce-based implementation of the algorithm.

Putting all the above together, I expect my frameworks to enable effective exploration and analysis of our KESM data sets. In addition, I expect my approaches to be broadly applicable to the analysis of other high-throughput medical imaging data.

1.3 Organization of the Dissertation

The rest of this dissertation is organized as follows. In Chapter 2, I discuss background information of my frameworks including the brain maps and atlases, machine learning, image noise removal, and spatial registration in bioinformatics. Chapter 3 introduces the Knife-Edge Scanning Microscope (KESM) and its data sets.

In the following chapters, I explain my approaches of exploration, registration, and analysis for the KESM data sets in details.

In Chapter 4, Chapter 5, and Chapter 6, I cover my exploration of the KESM data sets, presenting two versions of KESM Brain Atlas and a unit volume viewer to compensate the limitation of KESM Brain Atlas.

In Chapter 7, I deal with the registration of our KESM Nissl data set, introducing my 3D registration workflow to map the KESM data set into the Waxholm space.

In Chapter 8, I discuss the analysis of our KESM Nissl data set with a novel incremental learning approach for cell counting, showing superior results in comparison with current approaches.

Finally, Chapter 9 discusses the contributions of my frameworks and future work followed by Chapter 10 which concludes my dissertation.

2. BACKGROUND: RELATIONSHIP TO SIMILAR RESOURCES

Imaging technologies for generating massive, high resolution biological data sets are actively being developed. In this section, I survey existing data sources based on such technologies and algorithms for the visualization and analysis of such data.

2.1 Brain Maps and Atlases

A diversity of web-based high-resolution brain atlases are available. Below is a selected list of such web-based brain atlases.

The Allen Brain Atlas (ABA; `http://www.brain-map.org/`) provides web-based, publicly accessible datasets of over 20,000 genes expressed in the C57BL/6J mouse brain. The ABA archives millions of images from the mouse brain sectioned at a resolution of 0.95μm/pixel on the x- and y-axis and 25μm/pixel on the z-axis, and provides extensive annotation, which is semi-automatically generated by 3D registration. The ABA uses Adobe Flash web atlas in order to zoom and pan the data. The ABA provides human and non-human primate brain data sets. Recently, the ABA also added a high-resolution map of neural connections in the mouse brain [44, 38, 68, 90].

NeuroMorpho.Org is a neuroinformatics resource, providing thousands of digitally reconstructed neurons. It contains neuronal morphologies from different brain regions of rat, mouse, human, cat, monkey and so on. This portal provides a search service under diverse criteria such as categories, metadata, and morphometric measures. Searching by these criteria enables comparative morphological analysis [4].

Edinburgh Mouse Atlas Project (EMAP; `http://genex.hgu.mrc.ac.uk/intro.html`) uses optical projection tomography, a non-destructive scanning technique, to construct full 3D data of mouse embryos at the resolution of 2μm$\times$2μm$\times$2μm and

4μm$\times 4\mu$m$\times 7\mu$m on the order of hundred MB in size. Since their focus is mainly on embryonic studies, this medium-resolution suffices. Using the 3D datasets, EMAP presents 2D gene expression images in multiple orientations (transverse, sagittal, and coronal). They provide Java Atlas Browser using Java Applet to allow web access to the embryo datasets [7, 97].

BrainMaps.org contains digital brain atlases of various species including, but not limited to, primates, rodents, and avians. BrainMaps.org contains over 10 million megapixels of scanned data, with a typical resolution of 0.46μm/pixel with the section thickness of a minimum of 40μm. They provide AJAX online atlas, which enables zooming and panning of the pyramidal tiles of the image datasets, where label texts are overlaid on the image slides. For a 3D display, BrainMaps.org reconstructed some of the brain images into a VRML format, which requires a third party 3D display application [65].

The Mouse Atlas Project at UCLA (MAP; `http://map.loni.ucla.edu/`) contains an atlas viewer for human, monkey, and mouse brain datasets. They provide the option to choose amongst multiple volume data (an MRI volume, a block-face imaging volume, a Nissl-stained volume, and an anatomic delineation volume) sectioned at a coarse resolution of $70\times 50\times 70\mu\text{m}^3$ or $40\times 40\times 40\mu\text{m}^3$ for MRI volume and $1\times 1\times 50\mu\text{m}^3$ for the Nissl volume. Also, they provide an option to choose among different orthogonal orientations (transverse, sagittal, and coronal). However, the MAP atlas viewer does not allow navigating within an image (zooming/panning). There are many other atlases on the MAP archive, where the viewer is basically a "slice viewer," showing single-image planes at a time [48].

The Mouse Connectome Project (MCP; `http://www.mouseconnectome.org/`) at UCLA plans to collect high-resolution, multi-fluorescent connectivity atlases from the whole mouse brain using a double coinjection tracing method [32]. At their

project website, they allow users to access their connectivity data through a 2D interactive tool, iConnectome. Currently, they provide different regions of the mail olfactory bulb (MOB) connectivity data. Besides MOB data, they aim to gradually assemble the connectivity data of the accessory olfactory bulb (AOB), anterior olfactory nucleus (AON), and other olfactory cortical areas.

The Mouse Brain Architecture Project at the Cold Spring Harbor Laboratory (`http://mouse.brainarchitecture.org/`) aims to provide complete circuit mapping data in a whole mouse brain [66]. They used systematic injections of four different tracer types (two anterograde and two retrograde) at 262 grid locations in the mouse brain to characterize the circuits and cytoarchitecture. The project website allows users to access these preliminary data sets as well as auxiliary data sets with cytoarchitectonic markers.

The High Resolution Mouse Brain Atlas (HRMBA; `http://www.hms.harvard.edu/research/brain/index.html`) contains 572 coronal sections from C57BL/6J specimens. Each section is 20μm thick and treated in Nissl stain and Loyez method alternatingly. Consecutive image sections in a single stain are 40μm apart. They interpolated the consecutive sections to match the 40μm z-axis resolution to the 10μm x- and y-axis resolution. The 2D atlas of the HRMBA has an option to present each of the Nissl and myelin-stained sections separately or altogether. However, they do not allow navigational functions for the presented section images. They also reconstructed image sections for the cell clusters and grow dendrites to simulate axons and their synaptic connections in 3D.

Collaborative Annotation Toolkit for Massive Amounts of Image Data (CATMAID; `http://fly.mpi-cbg.de/~saalfeld/catmaid/`) is more focused on sharing of gigantic datasets. CATMAID uses the serial section Transmission Electron Microscopy (ssTEM) datasets of the drosophila first instar larval brains at a resolution

of 4nm/pixel. Inspired by GoogleMaps^{TM}, the lightweight CATMAID web interface allows decentralized image data storage and thus avoids duplication of massive datasets. Also, it provides basic navigational functions and allows collaborative annotation. Similar to the online atlas of BrainMaps.org, CATMAID also uses a pyramid of tiles for rapid browsing at multiple scales [77, 98].

The Biomedical Informatics Research Network project is a national initiative to share and mine data for biomedical research (BIRN; `http://www.birncommunity.org/`). This project's main goal is to create an integrated collaborative environment to share massive multiscale data collected across different advanced imaging modalities. Particularly, the Mouse BIRN provides magnetic resonance data (Duke), histological data (UCLA), diffusion tensor data (Caltech), and high-resolution electron tomography data (UCSD) for the mouse models. Also, a real-time multi-photon laser-scanning microscope at UCSD has generated multi-wavelength 4D datasets. To disseminate these huge multi-resolution datasets (that are multiple terabytes in size per data set) to biomedical researchers, the BIRN project is in need of a light-weight online interface [52, 19, 54, 53].

All of the above mentioned web-based atlases have attempted to solve the accessibility and visualization requirements. However, neither of the problems are sufficiently addressed. Not enough navigational functions are offered in some atlases (EMAP, MAP, and HRMBA), while some of them require additional software to be installed (ABA, EMAP, and BrainMaps.org). Moreover, none of them support a 3D view of the data at the resolution as high as their 2D versions. Visualization in 3D can greatly enhance the utility of these resources. However, the 3D views of ABA, BrainMaps.org, and HRMBA are too coarse, mainly because their datasets have poor z-axis resolution. Even though the ssTEM data that CATMAID uses is at the highest available resolution, at the nanometer scale [79, 78], 3D views are not

addressed.

More mouse brain atlases are available (not reviewed here), with data from different imaging modalities, but resolution for the whole mouse brain is not high enough to show neuroanatomical structures in detail.

2.2 Image Noise Removal

In medical imaging, noise suppression is an essential process to enhance the visibility of the objects of interest. A large number of techniques have adapted mathematical transforms for noise removal because such transforms help reduce noise while preserving structures in images. A variety of wavelet-techniques have been proposed and improved in the context of image processing to remove noise [101, 22, 9]. Starck *et al.* [88] employed the ridgelet and curvelet transforms to compensate for the limitations of wavelet methods that cause large wavelet coefficients at fine scales. Robinson *et al.* [75] extended a Fourier-wavelet deconvolution and denoising technique for efficiently generating high-quality images with enhanced contrast.

However, KESM has unique noise characteristics due to its use of physical sectioning coupled with simultaneous imaging, so a customized noise removal method is necessary. Particularly, tiny defects at the knife edge cause streaks. Moreover, the cutting process induces irregular marks due to vibration of the knife, which is called "chatter".

2.3 Brain Image Registration

Brain image registration is an important preprocessing step to investigate the intra-subject and inter-subject anatomical variability in brain mapping studies. With volumetric data sets exceeding a few terabytes like the KESM data sets, spatial registration into a standardized space is a further challenge.

The Talairach space, one of the most widely used common coordinate spaces for

the human brain, has provided a standard for describing locations of human brain structures with detailed anatomical labels in functional brain mapping studies [91]. In case of the mouse, Franklin and Paxinos [24] detailed stereotaxic coordinates in the three sectioning planes (coronal, horizontal, and sagittal) for the adult mouse brain atlas. These common coordinate spaces have allowed the alignment and comparison of various types of data acquired from different specimens.

The International Neuroinformatics Coordinating Facility (INCF) recently developed a canonical coordinate system for the rodent brain called the Waxholm Space (WHS) in 2008, for effectively integrating the rapidly growing rodent brain data. WHS is a rodent equivalent of the Talairach space for humans. WHS will provide facilities for standardization and comparison between rodent specimens [30].

So, to make our high-resolution and high-quality data broadly available and useful, I will register our Nissl volume data to the Waxholm space.

2.4 Machine Learning Approaches

Neuron counting serves as an important metric for early diagnosis of brain disorders, because the loss of neurons causes dysfunction in neuronal activities [95, 36, 26]. Hodneland *et al.* [33] introduced a unified framework for 3D segmentation of surface-stained living cells from fluorescent images. The authors used Hessian ridge enhancement and ridge enhancement by curvature for background subtraction. Based on the preprocessed images they ran watershed segmentation followed by level set segmentation for cell classification.

Jain *et al.* [35] emphasized the superior performance of machine learning for image segmentation. An initial algorithm takes images as input and when it produces segmentation output, they train the computer to learn to search a better algorithm that optimizes good performance. Jurrus *et al.* [39] used a series of Artificial Neural Net-

works (ANNs) to more accurately detect neuron membranes in Electron Microscopy (EM) images. At the beginning the first ANN uses raw image data as input, and then the following ANNs take the input vectors which the first ANN used, adding the output of the previous ANNs. [56] introduced a multi-layer feed-forward neural network model to perform cell detection in raw KESM data. To enhance the computational performance of their algorithm, they used constant memory on the Graphics processing units (GPUs).

These approaches assume that the intensity values of cell bodies are clearly distinct from backgrounds. Moreover, ANNs require a robust training set of manually classified samples. Due to these reasons, huge Nissl volume data from KESM, in which the shapes of cell bodies are diverse, is not able to simply use the current approaches.

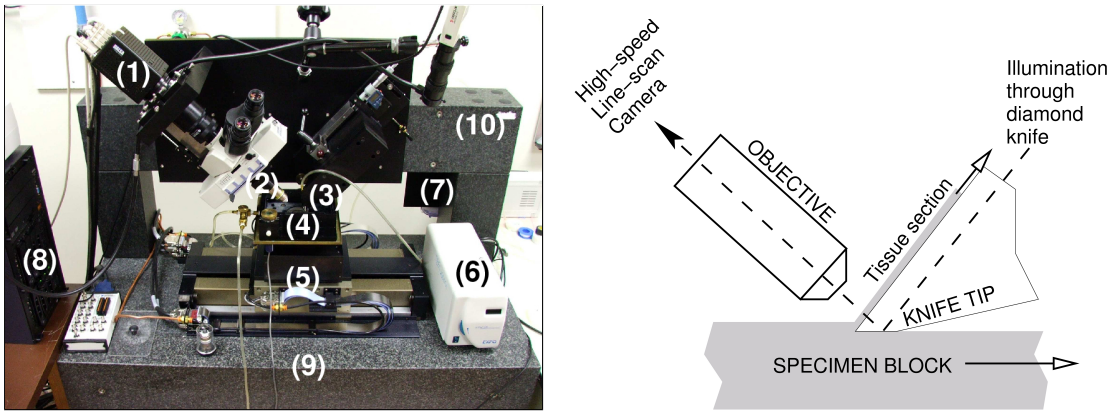
3. IMAGING WITH THE KNIFE-EDGE SCANNING MICROSCOPE

The Knife-Edge Scanning Microscope (KESM, US patent #6,744,572) [13, 43, 58, 62, 59, 61] has been designed at Texas A&M University (TAMU) in recent years with support from the National Science Foundation (MRI award #0079874; McCormick, PI), the Texas Higher Education Coordinating Board (ATP award #000512-0146-2001; Keyser, PI), and the National Institute of Neurological Disorders and Stroke (Award #1R01-NS54252; Choe, PI).

3.1 Knife-Edge Scanning Microscope (KESM)

The instrument, shown in Fig. 3.1A, is capable of scanning a complete mouse brain ($\sim 310\ mm^3$) at 300 nm sampling resolution within 100 hours when scanning at its full capability. The instrument comprises four major subsystems: (1) precision positioning stage, (2) microscope/knife assembly, (3) image capture system, and (4) cluster computer. The specimen, a whole mouse brain, is embedded in a plastic block and mounted atop a three-axis precision positioning stage. A custom diamond knife, rigidly mounted to a massive granite bridge overhanging the three-axis stage, cuts consecutive thin serial sections from the block.

Unlike block face scanning, KESM concurrently cuts and images (under water) the tissue ribbon as it advances over the leading edge of the diamond knife. A white light source illuminates the rear of the diamond knife, and in turn illuminates the brain tissue at the leading edge of the diamond knife with a strip of intense illumination reflected from the beveled knife-edge, as illustrated in Fig. 3.1B. The microscope objective, aligned perpendicular to the top facet of the knife, images the transmitted light. A high-sensitivity line-scan camera repeatedly samples the newly cut thin section, imaging a stripe 20 μm wide across the tissue ribbon and



A. KESM B. KESM operation

Figure 3.1: The Knife-Edge Scanning Microscope and its Operation. A. The Knife-Edge Scanning Microscope and its main components are shown: (1) high-speed line-scan camera, (2) microscope objective, (3) diamond knife assembly and light collimator, (4) specimen tank (for water immersion imaging), (5) three-axis precision air-bearing stage, (6) white-light microscope illuminator, (7) water pump (in the back) for the removal of sectioned tissue, (8) PC server for stage control and image acquisition, (9) granite base, and (10) granite bridge. B. The imaging principle of the KESM is shown. See [15] for details.

just beyond the knife-edge, prior to subsequent deformation of the tissue ribbon after imaging. Finally, the digital video signal is passed through image acquisition boards and stored in a dedicated cluster computing system. A custom software developed in-house at Texas A&M is used to control the stage movement and imaging process in a fully automated manner.

3.2 KESM Data Sets

We successfully imaged, using the KESM, entire mouse brains stained with Golgi (neuronal morphology), India ink (vascular network) [57], and Nissl (soma distribution) [14].

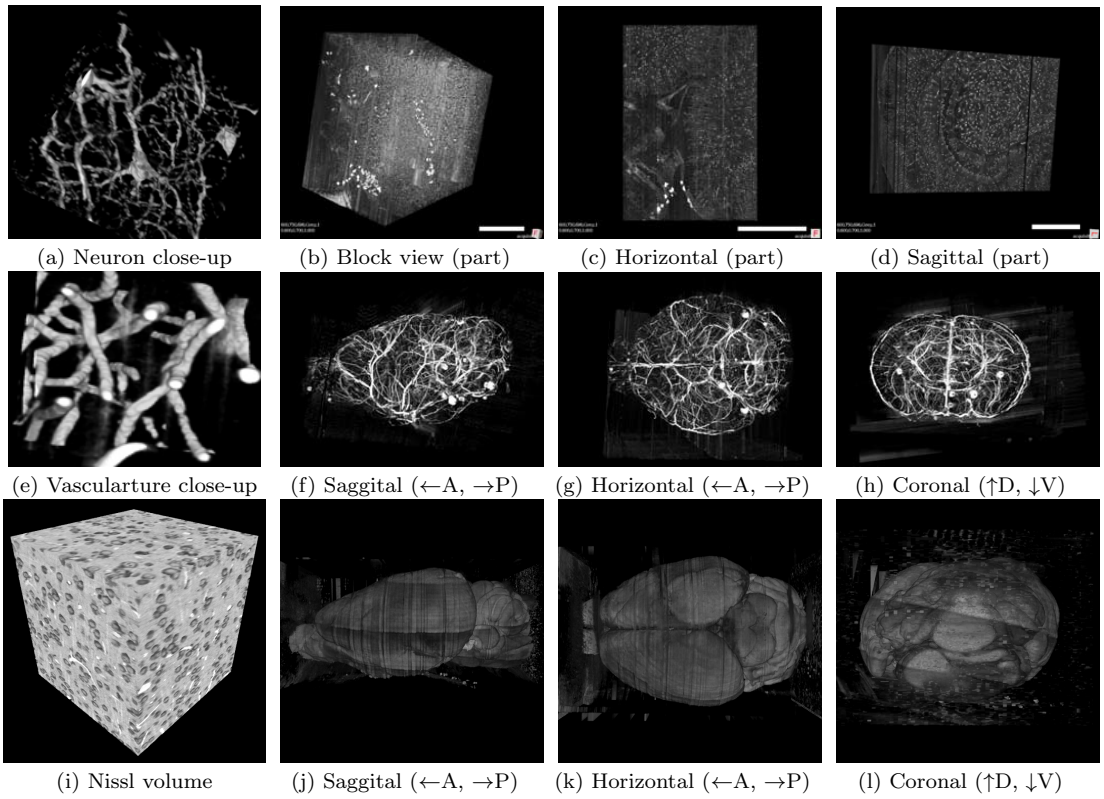
In 2008, we first imaged Golgi and India ink data sets from whole mouse (genotype C57BL/6J) brains. This Golgi data set did not include the left frontal lobe, part of

the left temporal lobe, and part of the right frontal lobe due to a misconfigured frame buffer that truncated the images, although the entire brain was sectioned using the KESM.

In 2010, we imaged Nissl and the second Golgi data sets from the KESM. Both data sets spanned the entire brain. The first Golgi data set, although partly incomplete, includes less noise than the second Golgi data set. Fig. 3.2 shows the screenshots of the KESM data sets. All data sets had a voxel resolution of 0.6 μm $\times$ 0.7 μm $\times$ 1.0 μm.

3.3 Summary

In this section, I introduced our Knife-Edge Scanning Microscope (KESM) and its data sets. Such a high-resolution mouse brain 3D volume data from KESM is a crucial resource in thoroughly understanding of the brain microstructures. Accordingly, in my dissertation, I will provide efficient and optimized frameworks for the unique KESM high-resolution data sets.



(a) Neuron close-up (b) Block view (part) (c) Horizontal (part) (d) Sagittal (part)

(e) Vascularture close-up (f) Saggital (←A, →P) (g) Horizontal (←A, →P) (h) Coronal (↑D, ↓V)

(i) Nissl volume (j) Saggital (←A, →P) (k) Horizontal (←A, →P) (l) Coronal (↑D, ↓V)

Figure 3.2: KESM Data. Volume visualizations of KESM data stacks are shown for the neuronal data set (top row, Golgi stain), the vascular data set (middle row, India ink stain), and the Nissl data set (bottom row, Nissl stain). (a) Pyramidal cells from the visual cortex. Width $\sim$ 100 μm (b) A large sub-volume from the Golgi data set. Fine details are washed out. (scalebar = 1.44 mm) (c) A thin slab from (b) reveals intricate circuits (horizontal section). (scalebar = 1.44 mm) (d) A thin slab from (b) reveals intricate circuits (sagittal section, scalebar = 1.44 mm). (e) Close-up of the vascular data. Width $\sim$ 100 μm. (f-h) Three standard views of the whole mouse brain vasculature (subsampled from high-resolution data, width $\sim$ 10mm). (i) Nissl-stained tissue volume ($\sim$ 300 μm^3). The dark donut-shaped objects are the cell bodies labeled by Nissl. White ovals are unstained regions representing blood vessels. (j-l) Three standard views of the whole Nissl mouse brain (subsampled from high-resolution data, width $\sim$ 10mm).

4. KESMBA V1: THE KNIFE-EDGE SCANNING MICROSCOPE BRAIN ATLAS USING GOOGLE MAPS API*

Connectomics is the study of the full connection matrix of the brain. Recent advances in high-throughput, high-resolution 3D microscopy methods have enabled imaging of whole small animal brains at a sub-micrometer resolution, potentially opening the road to full-blown connectomics research. One of the first such instruments to achieve whole-brain-scale imaging at sub-micrometer resolution is the Knife-Edge Scanning Microscope (KESM). KESM whole-brain data sets now include Golgi (neuronal circuits), Nissl (soma distribution), and India ink (vascular networks).

KESM data can contribute greatly to connectomics research, since they fill the gap between lower resolution, large volume imaging methods (such as diffusion MRI) and higher resolution, small volume methods (e.g., serial sectioning electron microscopy). Furthermore, KESM data are by their nature multiscale, ranging from the subcellular to the whole organ scale. Due to this, visualization alone is a huge challenge, before we even start worrying about quantitative connectivity analysis.

To solve this issue, I developed a web-based neuroinformatics framework for efficient visualization and analysis of the multiscale KESM data sets. The design requires that the framework is: (1) not dependent on high-end computer hardware (e.g., expensive graphics cards), (2) not dependent on custom 3D viewing applications or plug-ins, and (3) browsable within any standard web browser.

Our web-based neuroinformatics framework called KESM Brain Atlas (KESMBA)

*Reprinted with permission from "Multiscale Exploration of Mouse Brain Microstructures Using the Knife-Edge Scanning Microscope Brain Atlas" by Chung, Sung, Mayerich, Kwon, Miller, Huffman, Keyser, Abbott, and Choe, 2011. Frontiers in Neuroinformatics, 5:29, the Frontiers copyright line © 2011 under Creative Commons Attribution License.

v1 has been designed and implemented to allow the widest dissemination of KESM mouse brain data and to enable fast visualization.

4.1 Basic Idea: Transparent Overlay with Distance Attenuation

The basic idea I used to meet the requirements listed above is transparent overlay of images with distance attenuation [20]. Figure 4.1 shows the concept. Single images may not be informative (Figure 4.1A). Overlaying an image stack containing two intertwining objects to get minimum intensity projection (Figure 4.1B) results in the loss of 3D information. A good compromise would be to use transparency (alpha channel) in the stored images, so that clients can request multiple images from a stack and display it as an offset overlay to give a 3D effect (Figure 4.1C). This is similar to the artistic use of haze to achieve depth effect in a 2D medium (cf. [40]). In practice, raw images containing data already have semi-opaqueness in the background once made transparent, so simply overlaying them results in the same kind of effect. This simple approach, when combined with a Google Maps^{TM}-like zoomable web interface, results in a powerful browsing environment for large 3D brain data. In fact, we customized and extended the Google Maps API (version 2) to construct the KSEMBA.

4.2 Image Processing and Adding Transparency

With the raw image data sets acquired from the KESM, three additional image processing steps were performed to enhance the image quality suitable for the web atlas. First, to enhance visibility, I inverted the original images with black foreground and white background to have white foreground and black background. Next, because the inverted images do not have enough luminance contrast, I performed Gamma correction with a sigmoidal non-linearity to expand the luminance contrast between foreground and background pixels within each image. The pixel

A. Raw (synthetic) data slices

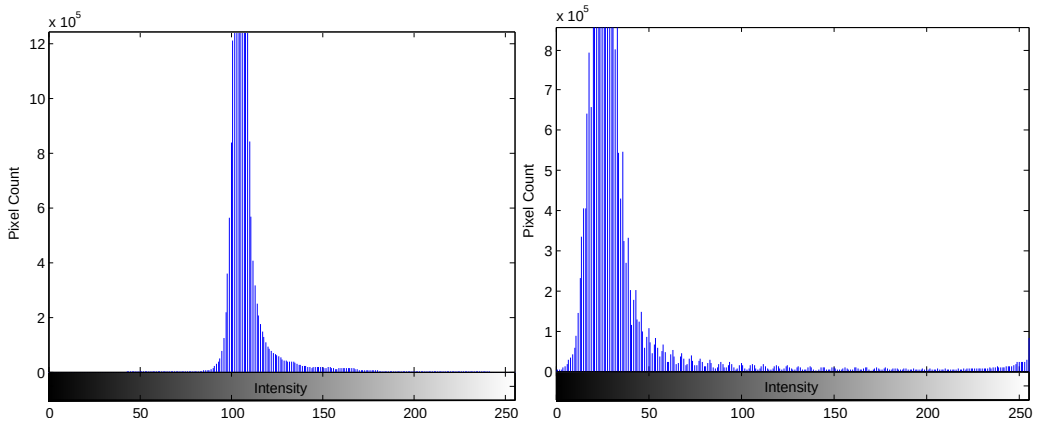
B. Overlay without distance attenuation C. Overlay with distance attenuation

Figure 4.1: Transparent Overlay with Distance Attenuation. A. An image stack containing two intertwined objects is shown. B. Simple overlay of the image stack in (A) results in loss of 3D perspective. C. Overlay with distance attenuation helps bring out the 3D cue.

count histograms in Fig. 4.2 show increased luminance contrast between foreground and background, with Gamma correction. Finally, I turned the background color of the image to be transparent, for a 3D view. One way to achieve image transparency on the web browser is to use the opacity feature of GoogleMaps APIs, which does not require further image processing. However, the opacity feature changes the transparency of all pixels in an image, so the foreground pixels that I want to keep opaque become as much transparent as the background pixels. Therefore, I performed one more image processing step to make the image pixels transparent according to their graylevel value. The processed images were stored in PNG format, which supports alpha channel transparency. The contrast factor and contrast center values (25 and 50) used in the graylevel transparency process were empirically selected. The following is the actual ImageMagick script for inversion, Gamma correction, and graylevel transparency:

```
convert <input_file.jpg> -negate \
-sigmoidal-contrast 25,50% \( -clone 0 \) \
-alpha off -compose copy_opacity - <output_file.png>
```

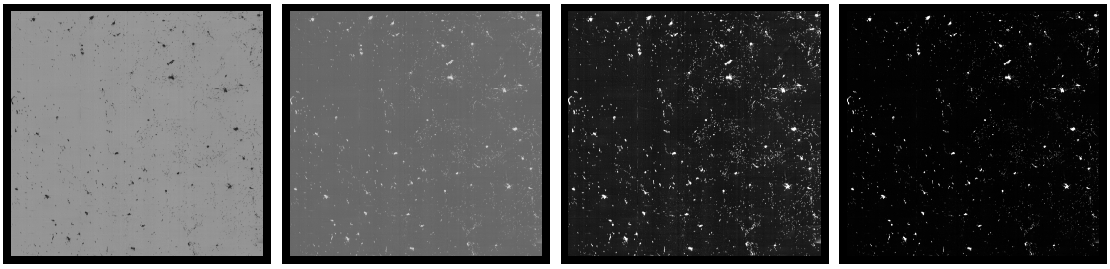
Results after each image processing step is shown in Fig. 4.3.



(a) Original (b) Gamma Corrected

Figure 4.2: Gamma Correction Result. Gamma correction is applied to a randomly selected image and the histograms of the graylevel intensity of (a) the original and (b) the Gamma corrected images are plotted. The foreground (near 160 intensity) and background (near 110 intensity) pixels are close together in the original. After Gamma correction, they are stretched to both ends.

Subsequently, pyramidal tiles, each of which is 256×256 pixel size, are generated by using ImageMagick. Each tile in GoogleMaps consists of pixels. The pyramidal structure of the GoogleMaps tiles in different zoom levels is shown in Fig. 4.4. Our Golgi data has 8 columns, each of which is 2,400×12,000 pixel images, making 19,200×12,000 pixels altogether. With them, I prepared tiles for 6 different zoom levels compatible with the GoogleMaps API's zoom level from 2 to 7. The number of tiles required at zoom level z is $2^z \times 2^z$. Therefore, the Golgi data set requires



(a) Original (b) Inverted (c) Gamma-Corr. (d) Transparent

Figure 4.3: Effects of Image Processings. Transformation of an image after each image processing step is shown.

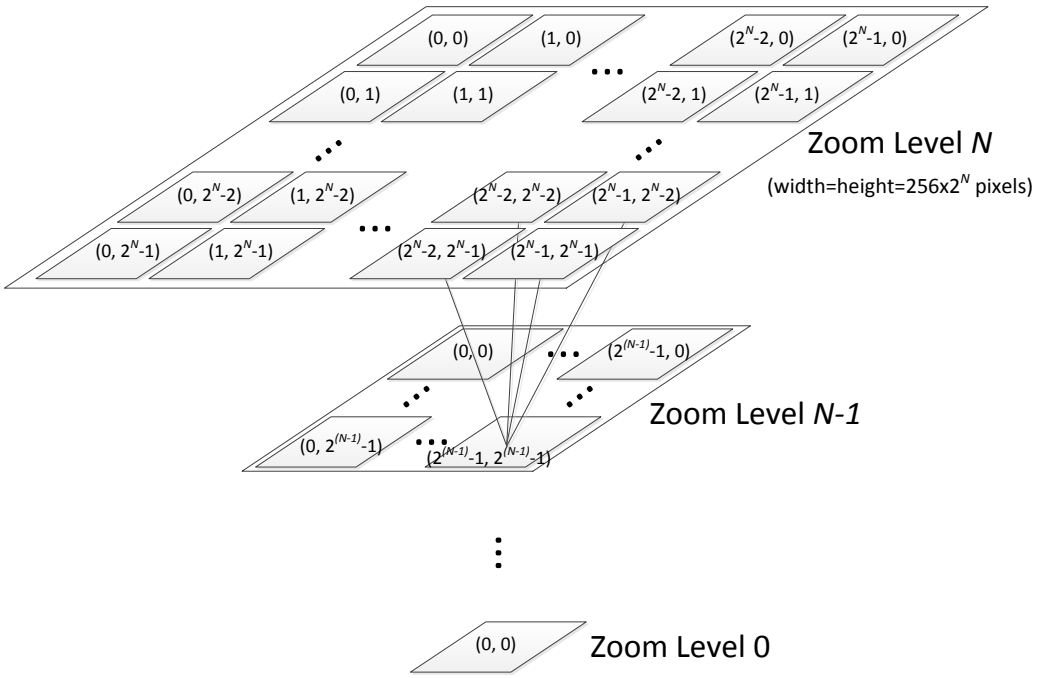


Figure 4.4: Tile Pyramid Compatible with GoogleMaps. Quad-tree pyramid of tiles and the xy coordinate indexing convention are depicted. Each tile has 256×256 pixels. Zoom level ranges from 0 to 17, and zoom level N has $2^N \times 2^N$ tiles. Following this tiling convention automatically enables various map functions including zoom in/out.

$\sum_{z=2}^{7} 2^z \times 2^z = 21,840$ tiles for each section, and 121,692,480 tiles for all 5,572 sections, theoretically. Fortunately, the actual number of tiles I created is 4,892 per

section and 27,258,224 overall because each image section is not square-shaped and I only had to create tiles containing tissue data. Fig. 4.5 shows the example of the tiles I created for the minimum zoom level 2 (i.e., lowest resolution). In the example, I had to create only 8 tiles out of 16 possible ones because I had no use for empty tiles. Preparing a tile pyramid requires extra storage, time and effort. Assuming that the above mentioned PNG transform did not increase the file size, in our Golgi data set, the tile pyramid ($4,892 \times 256 \times 256 \times 5,572 = 320,002,112$) causes about 39% increase in the file size compared to that without tiles ($19,200 \times 12,000 \times 5,572 = 230,400,000$). However, once they are generated, they contribute to saving image download time. For example, KESMBA v1 has a map area of

```
width:80\%;height:600px;
```

specified in a Cascading Style Sheet (CSS). The actual size of KESMBA v1 on clients' screens will be between 1536×600 and 820×600. This means that the number of 256×256 sized tiles concurrently displayed on a client web browser is only 28 at the maximum zoom level, which is less than 1% of the original image section (19,200×12,000 pixels).

Each tile is named to be consistent with GoogleMaps tile specification. For example, a tile name 1_2_3.png denotes zoom level=1, x-coordinate=2, and y-coordinate=3.

4.3 GoogleMaps-based Web Atlas

To enable 3D visualization, I customized the GoogleMaps API. GoogleMaps API provides extensive functions, required for a geographical atlas. In addition to the essential navigational functions of zooming and panning, GoogleMaps API offers useful features such as zoom scale bar, double-click zoom-in, overlaying various objects including image, text, marker, and polygon. GoogleMaps provides an extensive API specification and there exist a large number of private developers seeking and sharing

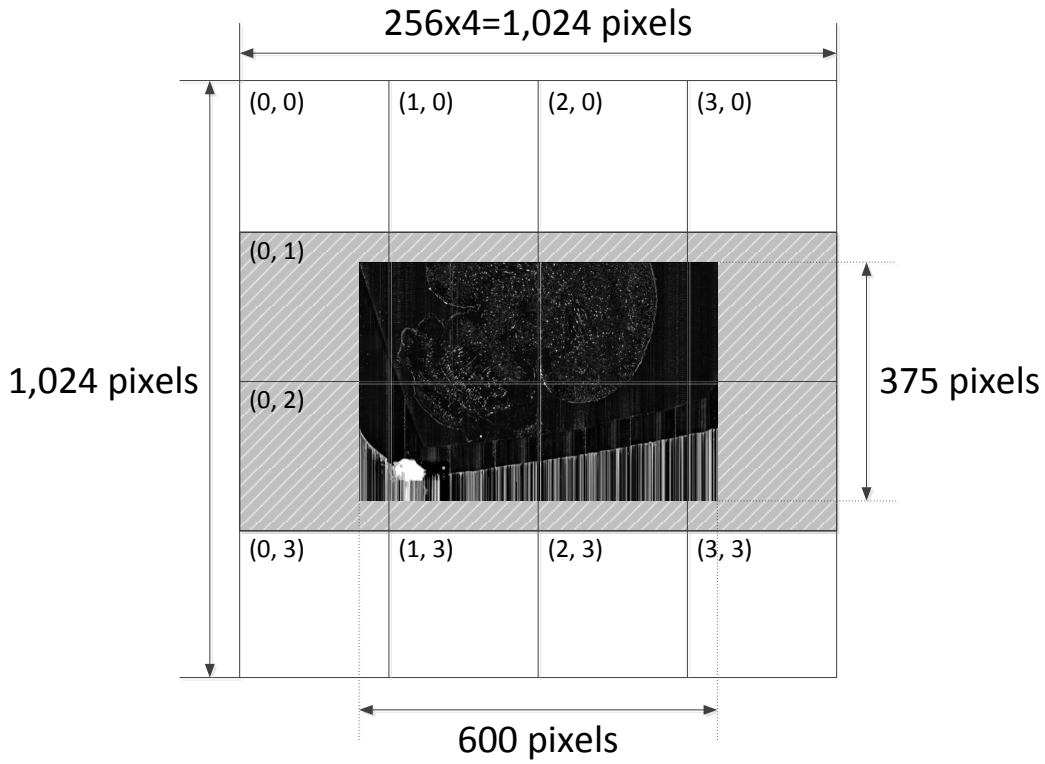


Figure 4.5: Example of Actual Tiles. The maximum zoom level of the Golgi image section (19,200×12,000 pixels) is 7 ($= \text{argmin}_x(256 \times 2^x \geq \max(19200, 12000))$). The minimum zoom level is set to 2. This particular example shows how the original image is tiled and how the tiles are named at zoom level 2. The dark image in the middle is the image section halved down 5 times (600×375 pixels) to fit in zoom level 2 (1,024×1,024 pixels). After putting the downsized image at the center, transparent image patches (gray dashed area) are added to fill the incomplete tiles so that every tile can have the same 256×256 pixels size. Because there is no need to generate empty tiles, only 8 tiles are created out of 16 possible ones.

solutions for customizing the API. Fig. 4.6 shows how I customized and extended the existing GoogleMaps API V2.

4.3.1 GoogleMaps Javascript API Customization

I generated a custom map type instance of the "GMapType" class to call the map tiles from the Google database to feed in the custom tiles we generated. Multiple tiles from subsequent image sections are overlaid to create a 3D effect. Below is the

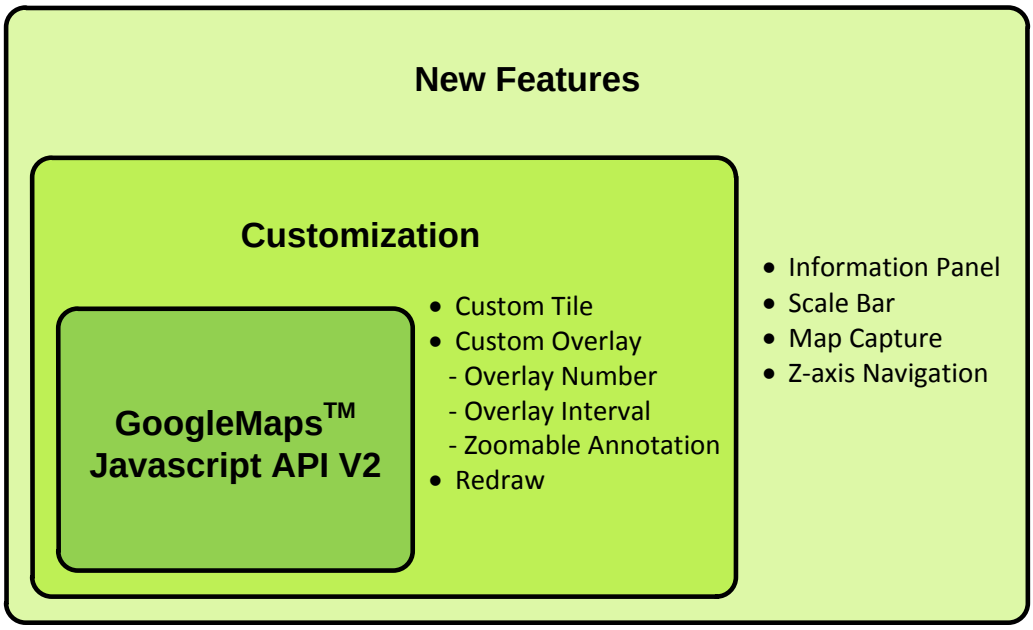


Figure 4.6: Extension of GoogleMaps API V2. Existing API functions are customized to fit the purpose of KESMBA v1. This customization includes: embedding custom tiles; tile overlays; user options to select the number and interval of the tile overlays; overlaying zoomable annotation; and map redraw function. The API is further extended to include: information panel; scale bar; map capture button; and z-axis navigation controller.

Javascript code for the map tile overlay customization:

```
[[overlay.js]]

...

// 1. Create a tile layer.

customLayer=[new GTileLayer(...)];

// 2. Generate a custom tile URL.

customLayer.getTileUrl=customGetTileUrl;

// 3. Create an overlay instance of the GTileLayerOverlay class.

customOverlay=new GTileLayerOverlay(customLayer);

// 4. Create a map type instance of the GMapType class.
```

```
customMap=new GMapType(...);
// 5. Create a map instance of the GMap2 class.
var myMap=new GMap2(document.getElementById("map"),
 {mapTypes:[customMap]});
// 6. Add predefined custom map type into the map instance.
myMap.addMapType(customMap);
// 7. Add a custom tile layer into the map instance.
cMap.addOverlay(overlays);
...
```

Users can select the number of tiles to overlay and the interval between two overlays, so that they can have freedom to generate the 3D view of their preference. On top of the image tile overlays, I added another optional overlay for text annotation. This annotation can change at different zoom levels, so that it can show more global description at a distant view and detailed description at close-up. Lastly, I added a "redraw" function so that when a user changes any of the above options (overlay size, overlay interval, and annotation on/off) and the map needs to be redrawn, it does not refresh the entire page but redraws only the map area. This is achieved by using the Javascript "arguments.callee" property, with which an executing function can recursively refer to itself. Below is the script for the map redraw function:

```
[[index.html]]
<body onload='load();'>


[[overlay.js]]
...
var loadFcn;  // global function pointer
```

```
function load() {
  map=newMap(arguments.callee, ..);
  // arguments.callee is the load() function itself
  ...
}
function newMap(caller, ...) {
  loadFcn=(caller) ? caller:function(){location.reload()};
  // Call loadFcn when the map needs to be reloaded.
  // caller is the original load() function
  ...
}
...
```

When "loadFcn" is called to redraw the map, it re-invokes the original "load" function. Redrawing the map in this way does not have to resume the entire page, and thus is faster.

4.3.2 Extended Features

Since the GoogleMaps API is created solely for 2D geographical maps, some features necessary for the 3D brain atlas are missing or do not allow user customization. I attached a horizontal menu bar on top of the map to include the main functions that are necessary for browsing KESMBA v1. Most of the new features are achieved by using various properties of the Document Object Model (DOM). In the menu, I added a z-axis navigation function with a drop-down menu (navigation step size) and buttons ("+" and "-" for moving in/out). Also, users can choose whether to display the annotation layer by clicking on a checkbox. To facilitate capturing the current

view on the atlas, I added an image capture button. When the button is clicked, it opens the print.html file using the windows.open method. In print.html, it gets the map area information of index.html by the "window.opener" property. Then, it copies whatever is on the map area of index.html to generate the page content of print.html using the "innerHTML" property:

```
[[index.html]]
...
<input id='print' type='button'
 onclick='window.open("print.html");' />
...
<div id='mapArea'></div>
...

[[print.html]]
...
<div id='container'>
  <script type='text/javascript'>
    var content=window.opener.document.getElementById('mapArea');
    document.write(content.innerHTML);
    //
  </script>
</div>
...
```

Since scale bar is one of the uncustomizable features of GoogleMaps, I attached the scale bar onto the map. I created a object using "createElement" method.

It was appended to the map div ( ) by using "getContainer" and "appendChild" methods. To make KESMBA v1 more informative, I also created a panel to display the information of the current view. In the panel to the right, KESMBA v1 displays the information about the specimen, stain type, current plane of view, dimension of the image section, and the z-range of the layers in the current view. An area to display the above information dynamically is first encapsulated by ... tags, and its contents are updated using the "firstChild" and "data" properties. This way, contents of the information panel are automatically updated as the user navigates or switches between the atlases using the top menu bar. Fig. 4.7 shows the interface of KESMBA v1 containing all the above mentioned features. For more detail, please see the caption for Fig. 4.7.

4.4 Results

Here, I will present the results of our two KESM Golgi data sets and a KESM India Ink data set from applying the KESMBA v1 framework to these data sets.

4.4.1 KESM Golgi Data Sets

The first Golgi brain was sectioned and imaged in 2008 (from July 7 to August 8, 2008). These results were first reported in [1]. The first Golgi data set did not include the left frontal lobe, part of the left temporal lobe, and part of the right frontal lobe due to a misconfigured frame buffer that truncated the images, although the entire brain was sectioned using the KESM. The second Golgi brain was sectioned and imaged in 2010 (from June 8 to August 4, 2010). The second data set contained the entire brain. The first Golgi data set, although partly incomplete, includes less noise than the second Golgi data set, so we decided to make available both data sets within the KESMBA v1 framework. These results are shown in Figures 4.8 to 4.10. All data sets had a voxel resolution of $0.6\mu m \times 0.7\mu m \times 1.0\mu m$, so at maximum

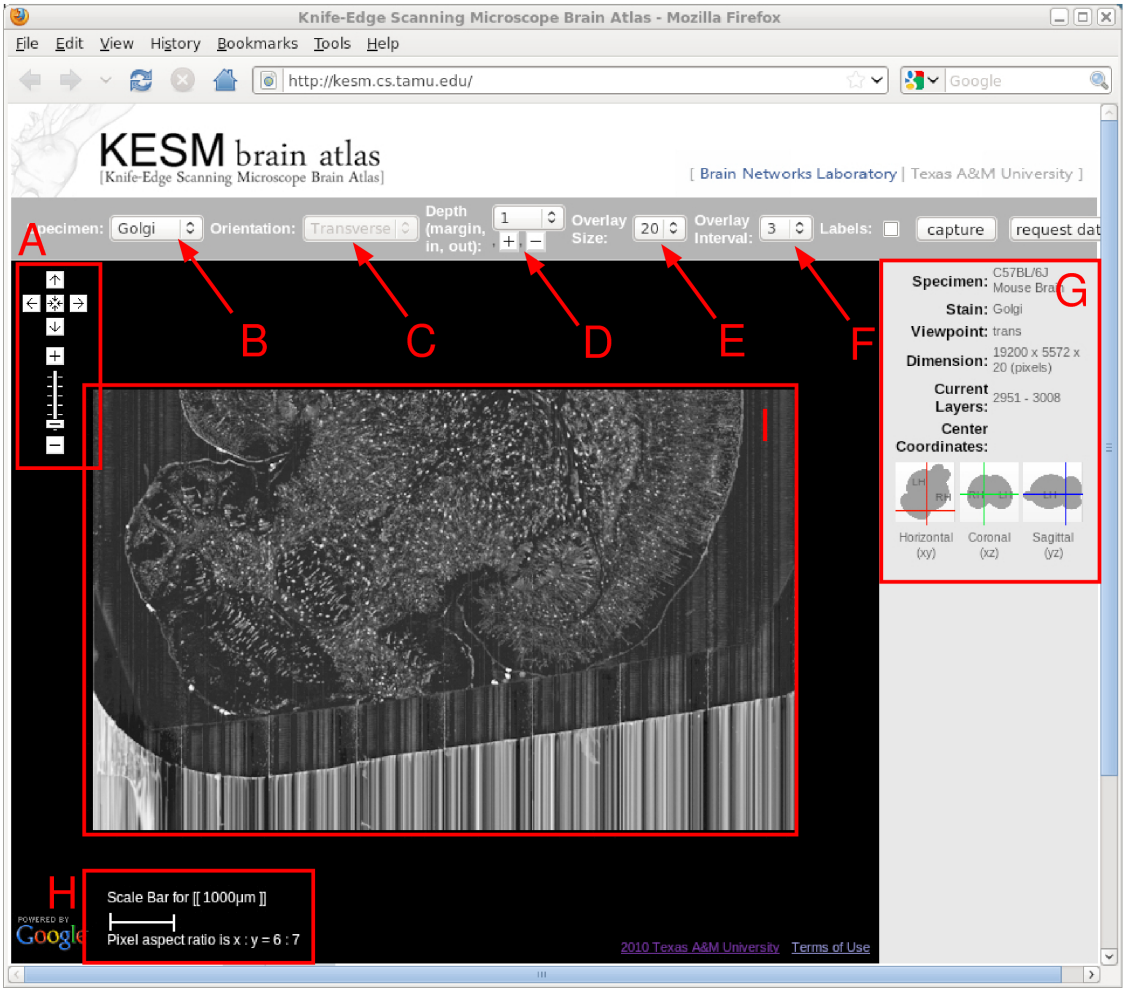


Figure 4.7: KESM Brain Atlas v1 Interface. A screenshot of KESMBA v1 running in a web browser is shown. Red markers and text were added on top for the purpose of explanation, below. A. Navigation panel: panning and zoom-in/zoom-out. B. Data set selection. Golgi, Golgi2, IndiaInk are available in the pull-down menu. C. Sectioning plane orientation. Three standard planes supported (planned). D. Depth navigation. Amount of movement (unit = 1 μm) in the z direction and forward (deeper, [+]) or backward (shallower, [-]) can be controlled. E. Overlay count. How many images to overlay can be selected here. F. Overlay interval. Overlaying evenly spaced at n μm intervals helps visualize thicker sections. G. Information window containing specimen meta data and current location information. H. Scale bar that automatically adjust to the given zoom level. I. Main display. Note that Google logo on the bottom left is shown due to the use of Google Maps API, and it by no means indicate any connection between the KESM data and Google.

zoom, the data are quite detailed, as shown in Figure 4.11.

4.4.2 3D Rendering through Image Overlays

All results shown in Figures 4.8 to 4.11 were from direct screenshots of KESMBA v1. The 3D effect is most notable in Figure 4.11. To highlight the z-axis resolution of the KESM data sets, and to show the effectiveness of our overlaying technique, we prepared views of a fixed region in the KESM data set by varying the number of overlays (Figure 4.12A–C). As we can see from this figure, overlays are effective in rendering 3D content, all within a standard web browser without any dedicated plug-in. Another technique that we implemented that is especially helpful when viewing with a larger field of view (i.e., zoomed out) is to overlay images at a certain interval. For example, overlaying 20 images at an interval of 5 would visualize a 100-μm-thick volume (compare Figure 4.12D and E).

4.4.3 Multiscale Nature of the KESM Data

One of the main advantages of KESMBA v1 is that it is very easy to navigate through the data, both within a certain scale and across multiple scales. In fact, this capability assisted greatly in producing the figures in the very article. Here, we will present the multiscale nature of the KESM data and show the effectiveness of the KESMBA v1 framework in handling such multiscale data. In Figure 4.13, we show successive snapshots of KESMBA v1 while zooming from the largest scale to the smallest scale. Each step of zooming in doubles the resolution, so the final panel has 32$\times$ higher resolution than the first panel.

4.4.4 Neuronal Circuits: Local and Global

Finally, we examine the relevance of the KESM data sets to connectomics research. Although it is true that with Golgi-Cox only $\sim$1 % of the entire population

of neurons are stained and thin myelinated axons are not stained reliably, we can still gain valuable insights from this whole-organ level data at a microscopic resolution.

KESM Golgi data sets can help advance connectomics research in two ways, (1) locally and (2) globally. At the local scale, we can investigate the basic circuits [83]. Although exact connectivity cannot be established, the repeating pattern can help us refine our basic circuit model, and also use the data to validate synthetic circuits constructed based on a theoretical generative model (see e.g. [42, 96]). Having access to these basic circuits from all regions in the brain is also a great benefit, as shown in Figure 4.14. This figure shows neurons from the cerebellum, inferior colliculus, thalamus, and hippocampus.

At the global scale, certain fiber tracts show up prominently in the KESM Golgi data. For example, various commissures in the frontal lobe and dense fiber bundles in the striatum are prominently visible (Figure 4.15). Similar fiber tracts can easily be identified, such as the hippocampal commissure in the posterior part of the brain.

4.4.5 Download Performance

The above results confirm the effectiveness of KESMBA v1's 3D view method using image overlays. However, the additional image overlays mean longer download time, and it will have limited utility if the download time exceeds waiting time tolerable for the users. Fig. 4.16 shows the result of download time analysis of KESMBA v1. Download time and download data size were measured in two modern web browsers (Internet Explorer 8.0.6 and Mozilla Firefox 3.6.8) using HttpWatch 7.0.26, a browser plugin to monitor http traffic. Expectedly, the download time and data size were proportional to the number of overlays. Notably, Firefox took extraordinarily long with large variance, while downloading 20 overlays. The Intranet and the Internet download times for 20 overlays reached above 22 seconds and 44

seconds respectively with Microsoft Internet Explorer, and 53 seconds and 52 seconds respectively with Mozilla Firefox. Literature on the tolerable waiting time for a web page download presents discordant thresholds between 4 and 41 seconds [82, 25], but none of them used a web page with as much graphical content as KESMBA v1. Considering the unique graphics-rich nature of KESMBA v1, we believe the above download times are within the tolerable threshold.

4.5 Summary

In this section, I presented KESMBA v1, a web-based mouse brain atlas using Google Maps API with improved accessibility and enhanced 3D visualization. KESMBA v1 was designed to facilitate the sharing of the massive high-resolution mouse brain data acquired from the KESM. The multiscale tiles allowed quick and consistent downloading time and the 3D method enabled effective 3D visualization. Moreover, KESMBA v1 allows access from any Internet devices because it minimizes the client-side computation and is implemented using standard Javascript library. KESMBA v1 can serve as an effective and efficient informatics framework for delivering large image volumes to the neuroscience research community.

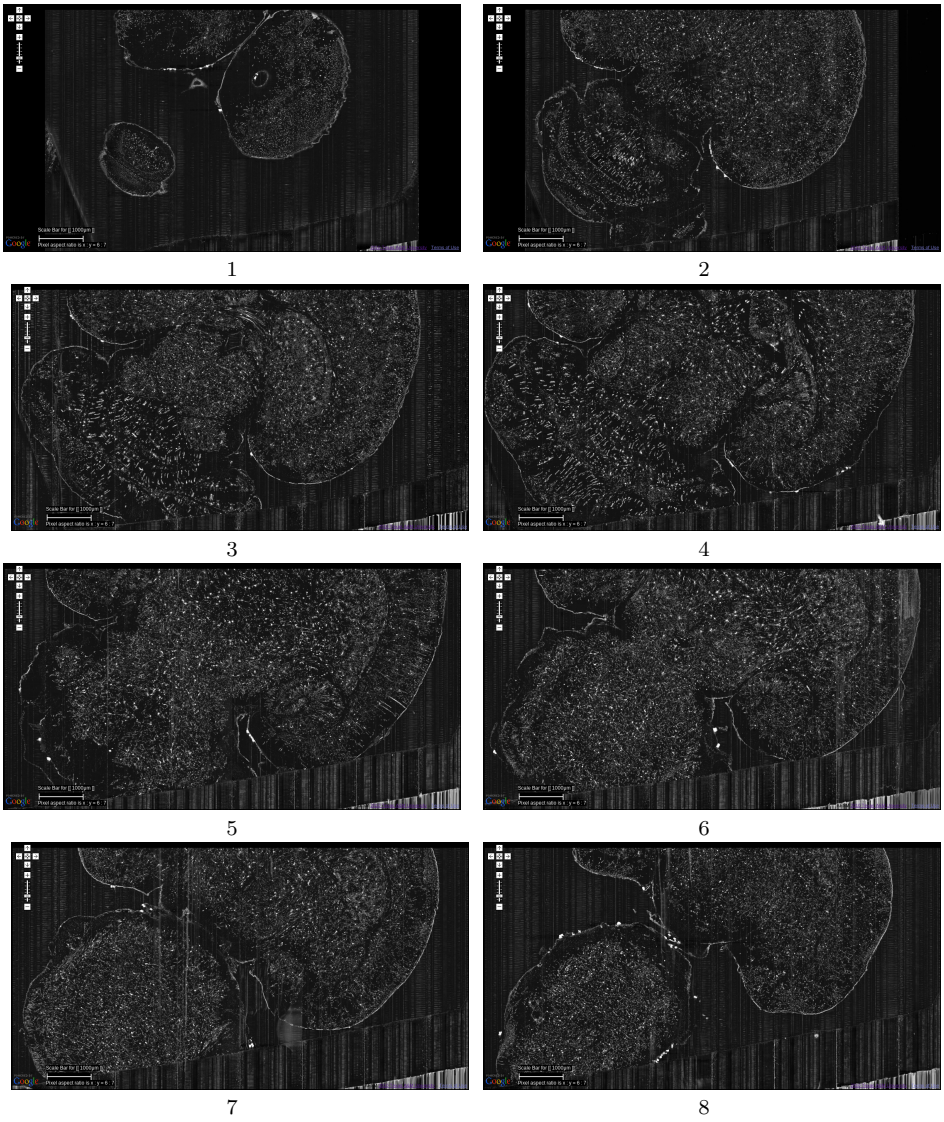


Figure 4.8: Golgi Data Set 1. A fly-through of the Golgi Data Set 1 is shown. The data were obtained by sectioning in the horizontal plane (upper right corner: anterior, lower left corner: posterior). This is the full extent of the data that was captured. We can see that part of the left temporal lobe, left frontal lobe, and part of the right frontal lobe are cut off. Scale bar = 1 mm. Each image is an overlay of 20 images in the z direction. The z-interval between each panel is 600 μm. The numbers below the panels show the ordering. These are cropped screenshots from KESMBA v1. This data set, obtained in 2008, is the first whole-brain-scale data set of the mouse at sub-micrometer resolution.

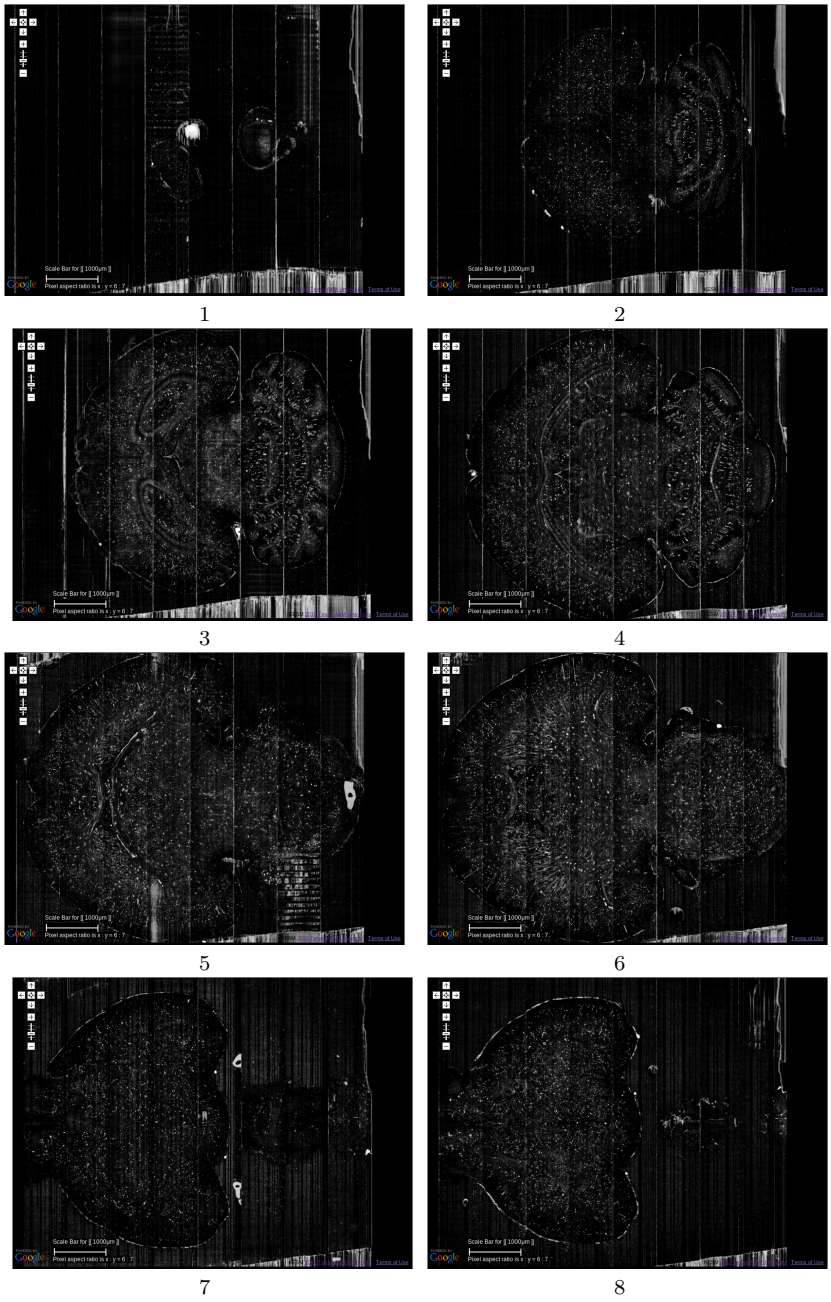
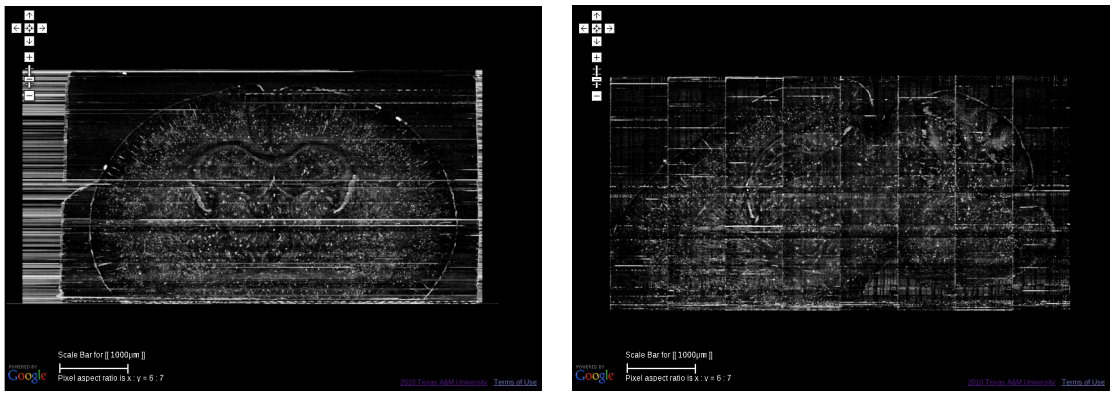


Figure 4.9: Golgi Data Set 2. A fly-through of the Golgi data set 2 is shown. The data were obtained by sectioning in the horizontal plane (left: anterior, right: posterior). Scale bar = 1 mm. Each image is an overlay of 20 images in the z direction. The z-interval between each panel is 800 μm, except for the last where it was 200 μm (so that data from near the bottom of the data stack can be shown: otherwise it will overshoot into regions with no data). The numbers below the panels show the ordering.



A. Coronal B. Sagittal

Figure 4.10: Golgi Data Set 2, Coronal and Sagittal Views. The coronal and sagittal views of the data set in Figure 5.5 are shown. Scale bar = 1 mm. These views show the superior z-axis resolution of the KESM data sets.

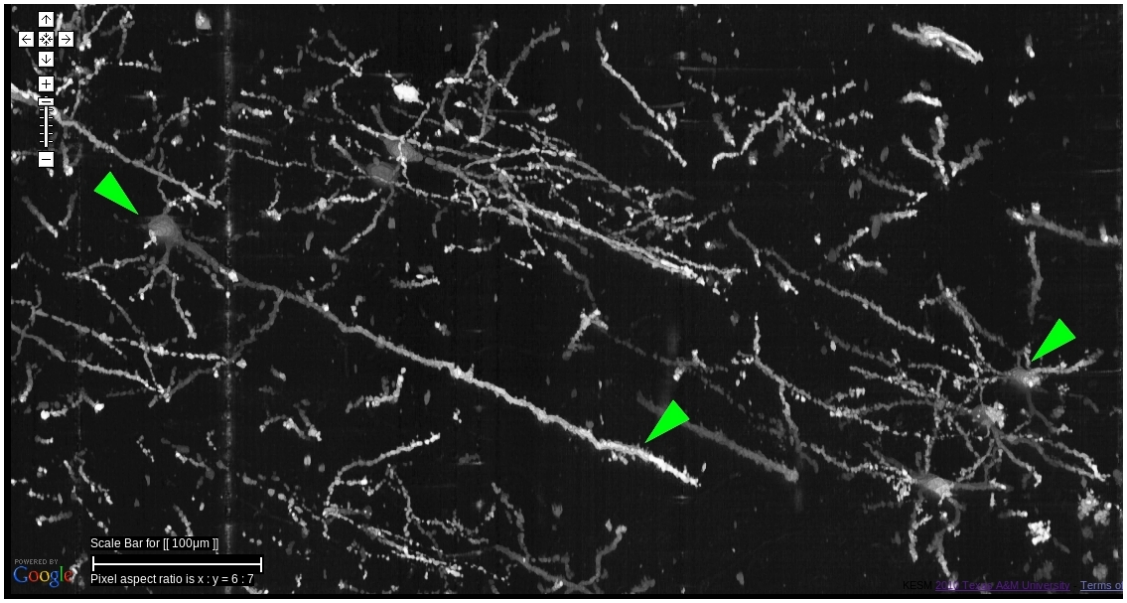
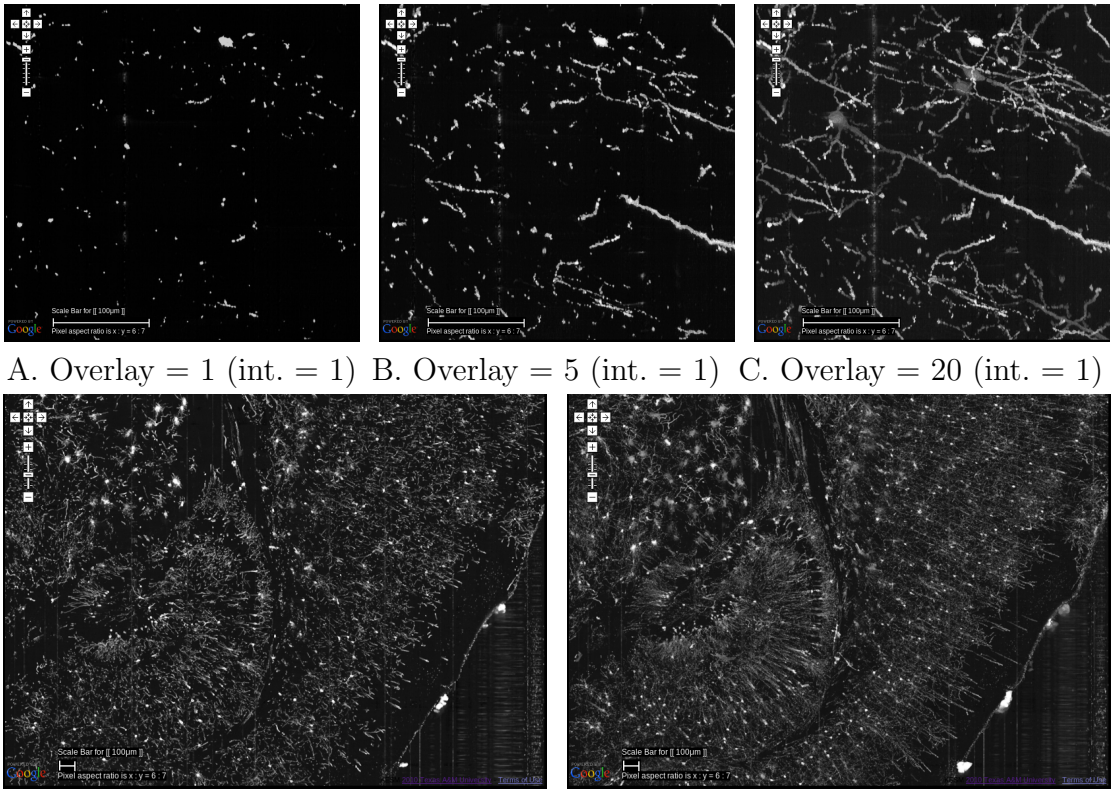


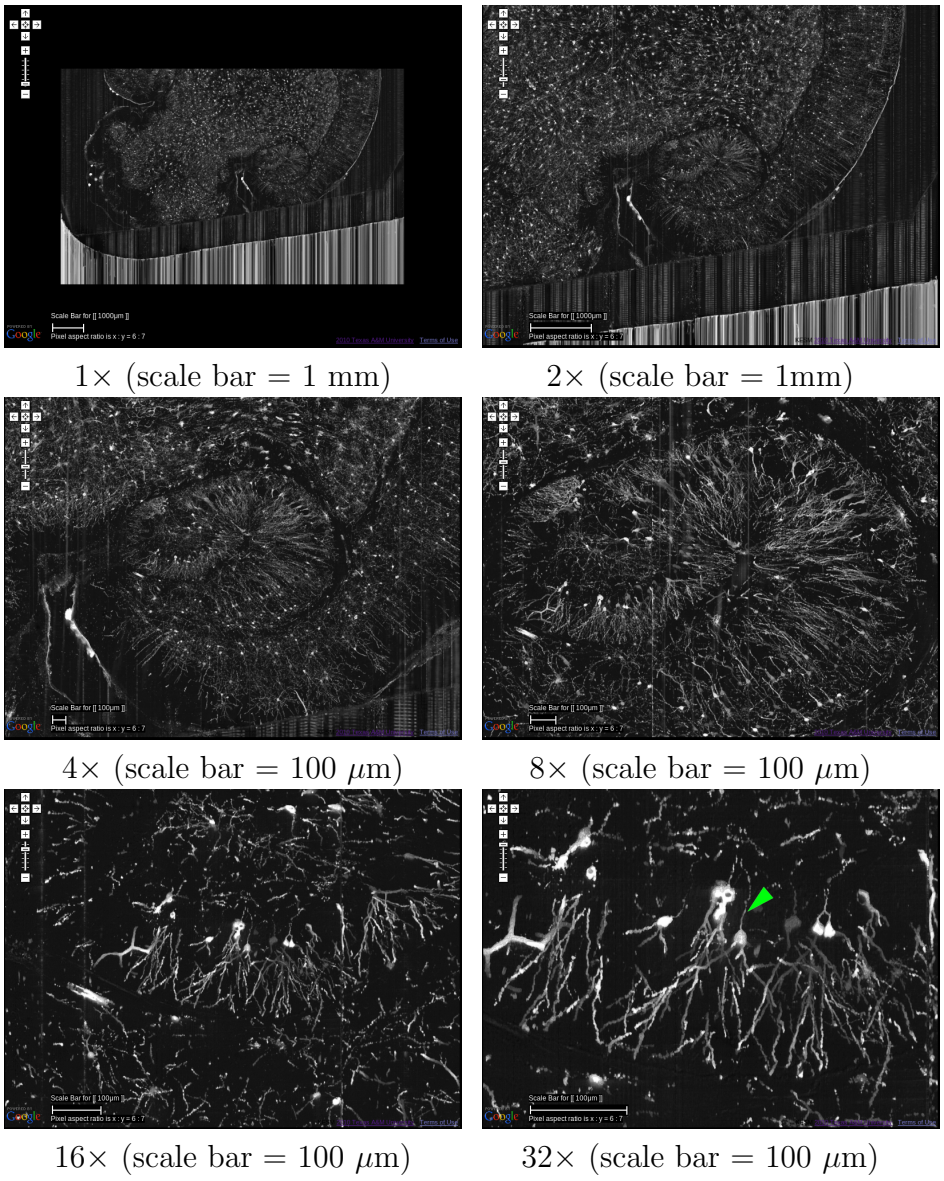
Figure 4.11: Details from Golgi Data Set 1. Details from the Golgi data set 1 are shown at full resolution. This panel shows an overlay of 20 images, thus it is showing a 20 μm-thick volume. Scale bar = 100 μm. The arrow heads, from left to right, point to (1) the soma of a pyramidal cell in the cortex and (2) its apical dendrite, and (3) a couple of spiny stellate cells. Other pyramidal cells and stellate cells can be seen in the background. At this resolution, we can see dendritic spines as well.



A. Overlay = 1 (int. = 1) B. Overlay = 5 (int. = 1) C. Overlay = 20 (int. = 1)

D. Overlay = 20 (int. = 1) E. Overlay = 20 (int. = 5)

Figure 4.12: Effectiveness of Image Overlays. The effect of an increasing number of overlays is shown. Scale bar is 100 μm and int. means interval. The data is from the same region as that from Figure 4.11. A. Since each KESM image corresponds to a 1-μm-thick section, a single image conveys little information about the neuronal morphology. B. Five overlayed images, corresponding to a 5-μm-thick section, begin to show some structure but it is not enough. C. With twenty overlayed images, familiar structures begin to appear. D–E. At a zoomed-out scale, skipping over images can be an effective strategy to view the circuits more clearly. In D, 20 overlays at an interval of 1, representing a 20-μm-thick volume, are shown. In E, 20 overlays at an interval of 5 is shown, representing 100 μm. The dense dendritic arbor in the hippocampus (left), fiber tract projecting toward the hippocampal commissure (middle, top) and the massive number of pyramidal cells and their apical dendrites (right) are clearly visible only in E.



1× (scale bar = 1 mm) 2× (scale bar = 1mm)

4× (scale bar = 100 μm) 8× (scale bar = 100 μm)

16× (scale bar = 100 μm) 32× (scale bar = 100 μm)

Figure 4.13: Multiscale View of KESMBA v1. A multiscale view of KESMBA v1 is shown (Golgi data set 1), by gradually zooming into the hippocampus (the numbers below the panels show the zoom-in sequence). All panels show an overlay of 20 sections. The first four panels are shown with an overlay interval of 5 and the last two with an interval of 1. Axons emerging from the hippocampal neurons are clearly visible (arrow head, last panel).

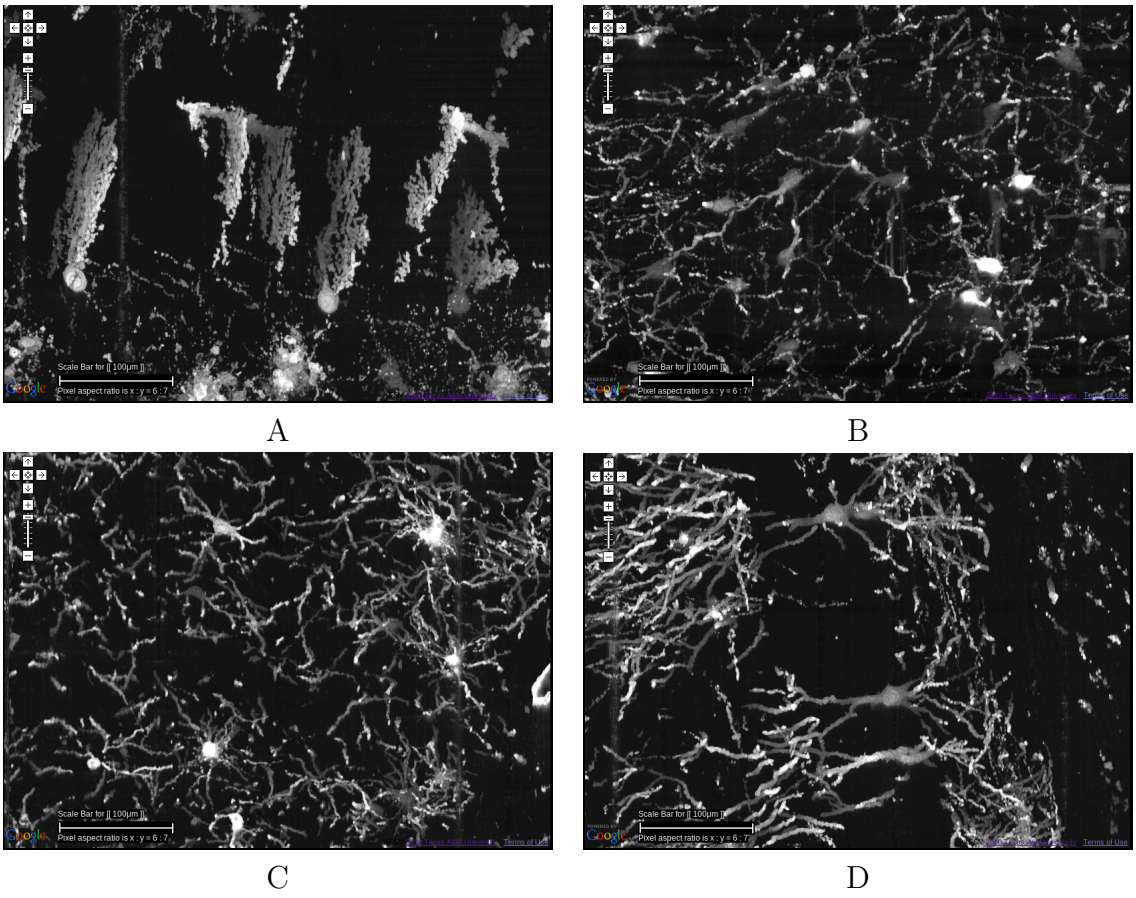
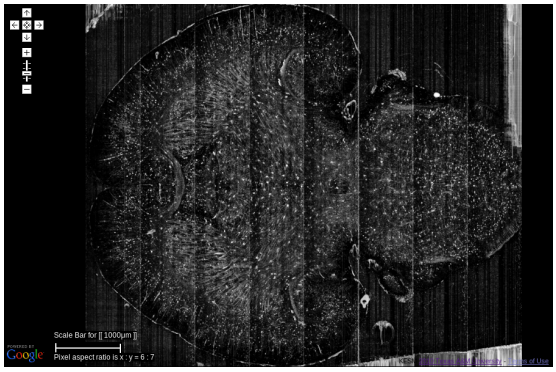
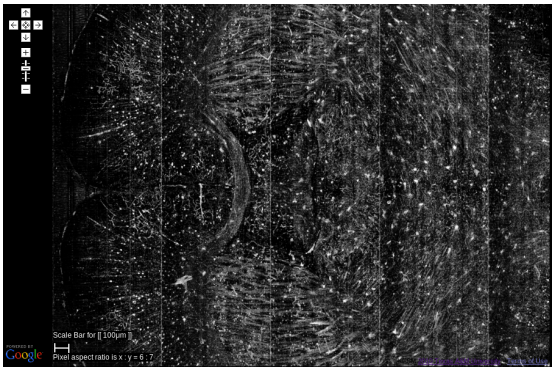


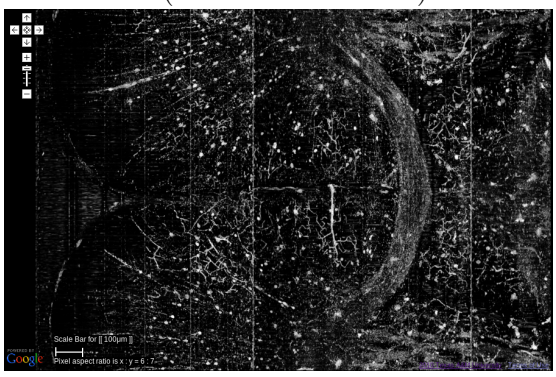
Figure 4.14: Different Types of Local Circuits. Different types of local circuits from the KESM Golgi data set 1 are shown. A. Cerebellum. B. Inferior colliculus. C. Thalamus. D. Hippocampus (also see Figure 4.13). See Figure 4.11 for circuits in the neocortex. Scale bar = 100 μm.



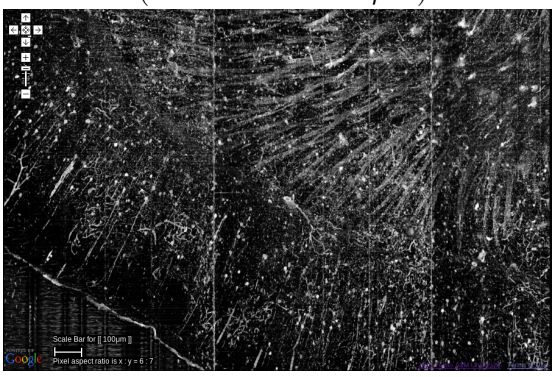
A (scale bar = 1 mm)



B (scale bar = 100 μm)



C (scale bar = 100 μm)



D (scale bar = 100 μm)

Figure 4.15: System-Level Fiber Tracts in the KESM Golgi Data Set 2. A. Horizontal section at the level of the anterior commissure (the ")"-shaped fiber bundle) is shown (left: anterior, right: posterior). Massive fiber tracts in the striatum can also be observed. B&C. Zoomed in view showing the anterior commissure near the middle. D. Close-up of the fiber bundles in the striatum can be seen. A large number of apical dendrites in the adjoining cortex can also be seen.

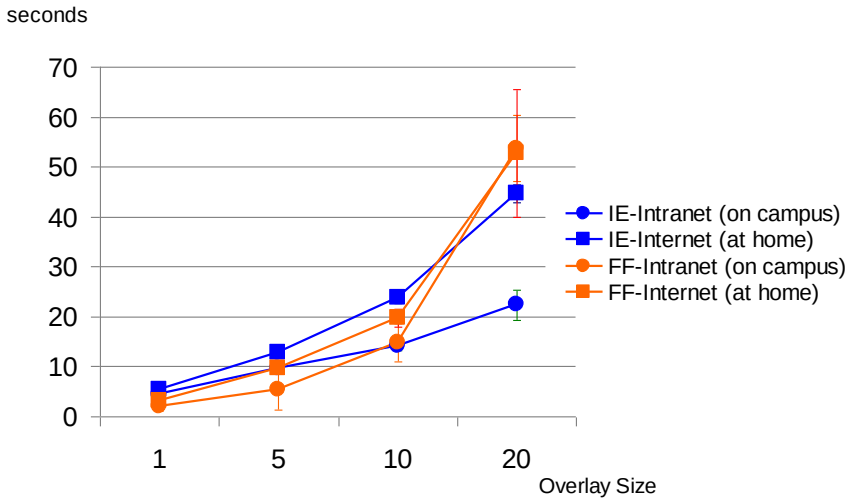


Figure 4.16: KESMBA v1 Download Performance Analysis. Average of 10 download trials for each setting (browser type and overlay size) is plotted (error bars indicate standard deviation). IE = Internet Explorer, FF = Firefox. Except for the case of Mozilla Firefox downloading 20 overlays, both the Intranet and Internet downloading times increased proportionally with the overlay size.

5. KESMBA V2: THE KNIFE-EDGE SCANNING MICROSCOPE BRAIN ATLAS USING OPENLAYERS API

To disseminate our high-resolution whole mouse brain data sets to the neuroscience research community, I designed and implemented a web-based brain atlas using Google Maps Javascript API (KESMBA v1 `http://www.kesm.org/`). KESMBA v1 provides 3D visualization by customizing Google Maps API, not requiring high-performance graphics cards or any web browser add-ons.

However, because Google Maps API is governed by a commercial license, customization, extension, and mirroring of the library remain limited. Furthermore, Google Maps API requests an API key from Google for every single page view, so KESMBA v1 needs to be connected the networks at all times. Moreover, the added alpha channel to image tiles for transparency to create a 3D effect increased the size of each tile images. Because of the increased size of each tile image, multiple overlays lead to the loading time being very long (e.g., loading to overlay 50 images takes more than one minute). KESMBA v1 also only supports the native orientation of the data sets: the Golgi data sets were obtained in horizontal sections; and the India ink and Nissl data sets were in coronal sections. Restricting to a single orientation hampers exploration and understanding of the data.

To build a better web-based neuroinformatics framework, I adapted an open source JavaScript library for displaying map data, the OpenLayers API. OpenLayers API supports geographic navigation available in Google Maps API such as panning, zooming, and overlaying. Because this API uses an open-source license, I can directly download the recent version of the OpenLayers library at `http://openlayers.org/`, customize the library, and adapt extensions of other developers. It does not require

API key, so I can run this framework locally. I call this framework KESM Brain Atlas (KESMBA) v2.

Furthermore, to reduce page loading time, I decreased the file size of the tile images by converting our bitmap-based mouse brain data sets to XML-based vector graphics format, Scalable Vector Graphics (SVG), while preserving the information content. On top of that, to minimize bringing up unit volumes for fully interactive 3D inspection, a good alternative is to provide all three standard orientations: horizontal, coronal, and sagittal. KESMBA v2 has a pull-down menu to choose between these three standard orientations.

KESMBA v2 allows open-ended innovation and an offline service. It also enables much faster and more orientations of visualization compared to KESMBA v1.

5.1 Image Processing and Converting Bitmap to Vector Graphics

We used the KESM for sectioning and imaging tissue blocks of the mouse brain. The actual imaging was performed by a DALSA CT-F3 high-sensitivity line-scan camera capturing the transmitted light, and these images were stored in the designated storage. Noise due to the knife-edge misalignment, defects in the knife blade, and knife chatter were removed through image processing algorithms including light normalization [58]. The KESM controller also employed a stair-step cutting algorithm because of limited field of view and width of the knife [43]. After preliminary image processing for noise and distortion removal, TIFF formatted original raw files were compressed into high quality JPEG format and stored for further processing. We imaged horizontal sections from the Golgi stained and coronal sections from the India ink stained brain.

Once we prepared JPEG image data, I reduced the graylevel images to binary image using Otsu's threshold clustering algorithm [69]. Comparing to other threshold

clustering approaches such as maximum entropy, isodata, and mean, Otsu's threshold method preserved as much as the tissue structure of raw image data. Once the binary images were generated, multiscale tiles were generated, each of which consisted of 256×256 pixels as in KESMBA v1.

To generate multiscale tiles from the binary image data sets, first I calculated the number of possible tiles along both x- and y-axes by dividing the each image size by the tile size 256×256 pixels. If the number of tiles along each axis was not even, I made the number of tiles even by adding one. Then, I calculated the number of possible zoom levels by applying the logarithm of base 2 to each tile count. Next, I calculated the number of required tiles at the maximum zoom level. If the number of tiles is less than the required number, the differences between the number of tiles and the required number along each axis was used to compute the number of padding tiles around the data tiles. The following Python code describes this computation. This is the code for generating the tiles from the original image data and for the tiles of the down-scaled images. I scaled down the raw images first and generated the corresponding tiles in the same manner. Each tile was named following the Google Maps tile naming convention because OpenLayers also supports this convention (e.g., 1_2_3.png showing zoom level = 1, x-coordinate = 2, and y-coordinate = 3).

```
////////////////////
// File: tile_gen.py
////////////////////
...
// Calculate the number of possible tiles
tile_x_num = math.ceil(img_x_pix / 256)
tile_y_num = math.ceil(img_y_pix / 256)

// Make tile numbers even
```

```
if(tile_x_num % 2 != 0): tile_x_num += 1
if(tile_y_num % 2 != 0): tile_y_num += 1

// The number of possible zoom level
zoom_num = math.ceil(math.log(tile_x_num,2))

// The number of required tiles at the maximum zoom level
max_tile_num = math.pow(2,zoom_num)

// The number of pad tiles
pad_x_num = max_tile_num - tile_x_num
pad_y_num = max_tile_num - tile_y_num
...
```

Once I had prepared PNG tiles, I used Potrace [81] to generate Scalable Vector Graphics (SVG) vector image data from the multiscale bitmap tiles. Potrace is an open source tool which transforms bitmaps into vector graphics. Compared to Autotrace (`http://autotrace.sourceforge.net/`) – an open source vector-converting tool, Potrace produces superior graphical outputs, reducing processing time and file size. See [81] for a detailed comparison between the Potrace and Autotrace.

I used Potrace version 1.11 (`http://potrace.sourceforge.net/`), and changed *backend_svg.c* code in the Potrace source distribution, which generates SVG files from the bitmap tracing results. In the original code, the *backend_svg.c* code gets the width and height of bitmaps, and using the information, sets the width and height of the SVG format. However, because the code adds *pt* unit, which measures the height of a font, after the values of the width and height, the size of the original view is changed by the point unit. The code below solves this problem by taking away the *pt* unit, that is, it assumes the unit is a pixel.

```
////////////////////
// File: backend_svg.c
////////////////////
...
// The original code line
// fprintf(fout, " width=\"%fpt\" height=\"%fpt\" viewBox=\"0 0 %f %f\"\n",
//             bboxx, bboxy, bboxx, bboxy);
// The updated code, which removed pt units.
fprintf(fout, " width=\"%f\" height=\"%f\" viewBox=\"0 0 %f %f\"\n",
               bboxx, bboxy, bboxx, bboxy);
...
```

When I transformed the bitmap tiles to SVG vector image data using Potrace, the format of our bitmap tiles was BMP, because Potrace only supports PBM, PGM, PPM, or BMP format. Potrace provides multiple options for better tracing of bitmap images such as suppressing speckles, curve optimization, and corner thresholding. It also has the options of generating SVG files such as grouping related paths and a single path for whole image. I only used the option of suppressing speckles in the Potrace and traced our bitmap tiles, converting them to SVG data. Here is the command line in the Linux system when I run Potrace for the conversion.

```
// -t n        - suppress speckles of up to this size (default 2)
// -s          - SVG backend (scalable vector graphics)
// -o filename - write all output to this file
>potrace tile_input.bmp -t 0.1 -s -o svg_output.svg
```

5.2 Web Atlas Based on OpenLayers

To enable 3D visualization, I customized the OpenLayers Javascript library released as open-source software. OpenLayers provides an extensive range of functions

required for geographic information rendering. Besides the main navigational functions of zooming and panning, OpenLayers supports useful features such as zoom scale bar, double-click zoom-in, and overlaying various objects including images, text, markers, and polygons. On top of these functions, the open library can be directly edited and customized by the user.

5.2.1 Customizations

Existing library functions were customized to fit the purpose of KESMBA v2. This customization included the use of custom SVG tiles, tile overlays, user options to select the number and interval of the tile overlays, and overlaying zoomable annotation. The library was further extended to include information panel, scale bar, and z-axis navigation controller.

OpenLayers consists of two components: map and layer. An instance of the OpenLayers map stores basic information about the boundary, the number of zoom levels, the default projection, units, and controls of the map frame. Data information is displayed by adding a layer to the map instance. To create a map instance, I used the `OpenLayers.Map` constructor. OpenLayers provides different subclass groups of Layer such as Web Map Service (WMS), Tile Mapping Service (TMS), Vector and so on. I used the `OpenLayers.Layer.TMS` constructor to create a layer because the TMS constructor supports SVG tiles by designating the tile image `type` 'svg'.

Inside the Map instance, the first layer instance is the base layer for the map frame. On top of the base layer, multiple layers from subsequent image sections can be overlaid to create a 3D effect. The code summary below provides an overview of how this is achieved using OpenLayers.

```
////////////////////
// File: svglayers.js
```

```
////////////////////

...

// 1. Create the map frame.
g_map = new OpenLayers.Map( ‘svgmap’, {
        maxExtent: new OpenLayers.Bounds(-80150033.3568,-80150033.3568,
                                80150033.3568,80150033.3568),
        numZoomLevels:g_zoom,
        maxResolution:156543.0339,
        units:‘m’,
        projection: ‘‘EPSG:900913’’,
        displayProjection: new OpenLayers.Projection(‘‘EPSG:4326’’),

        controls: [
            new OpenLayers.Control.PanZoomBar(),
            new OpenLayers.Control.Navigation(),
            new OpenLayers.Control.Attribution(),
            new OpenLayers.Control.KeyboardDefaults()
        ]
    });

// 2. Add the base layer to the map frame.
var new_url = g_url_add + fiveDigitize(g_curr_depth) + ‘‘/’’;
var b_layer = new OpenLayers.Layer.TMS(‘‘BaseLayer’’, [new_url], { type:‘svg’,
                                getURL:get_kesm_url, opacity: g_valOpacity});
g_map.addLayer(b_layer);

// 3. Add more layers to the map frame.
if(g_numOverlay > 1)
{
```

```
    var overlays=[];
    for(var idx = 0; idx < g_numOverlay-1; idx++) {
      new_url = g_url_add+fiveDigitize(g_curr_depth+g_intOverlay*(idx+1))+‘‘/’’;
      overlays[idx] = new OpenLayers.Layer.TMS(‘‘Name’’, [new_url], { type:‘svg’,
          getURL:get_kesm_url, opacity: g_valOpacity,
          isBaseLayer: false, transparent: true });
      g_map.addLayer(overlays[idx]);
    }
}

...
```

In my case, users can generate a 3D view of volumetric data by overlaying different number of layers and by adjusting the interval between successive image sections. On top of the data layers, users can also add an annotation layer.

The OpenLayers has a scale bar library for topography data and the constructor of this library is `OpenLayers.Control.ScaleBar`. I adapted this library to add the scale bar feature to our brain atlas. In the library source code, I changed the measurement property to match that of our brain atlas. At the zoom level 0 (i.e., the original data resolution) the 100-pixel scale bar corresponds to 100 μm in the brain tissue and at successfully higher zoom-out levels the corresponding length of the tissue doubles. The modified code is shown below.

```
/////////////////////////////////////
// File: OpenLayers-2.12/lib/OpenLayers/Control/ScaleBar.js
/////////////////////////////////////

...
```

```
 measurementProperties: {
        kesm: {
            units: ['micrometers', 'millimeters', 'micrometers'],
            abbr: [eval("String.fromCharCode(956)")+'m', 'mm', 'm'],
            inches: [393700.7874, 39.370079, 0.393701]
        }
    },

...
```

In order to generate an instance of the scale bar and add the scale bar instance to the map frame, I added the code below in the map initialization code.

```
////////////////////
// File: svglayers.js
////////////////////

...

// scale bar
var scalebar = new OpenLayers.Control.ScaleBar();
scalebar.divisions = 1;
scalebar.subdivisions = 1;
scalebar.singleLine = true;
scalebar.abbreviateLabel = true;
scalebar.displaySystem = "kesm";
// g_map is the map frame instance
g_map.addControl(scalebar);

...
```

5.2.2 Extensions

I also added new features for the 3D brain atlas which are not provided by OpenLayers. A z-axis navigation feature was added that could be controlled by a drop-down menu (for selecting the next depth size) or two keys ("-" and "+" for moving backwards/forward). Users can also activate an annotation layer by checking a checkbox.

Besides these brain atlas features, I created an information panel to display the status of the current view. This panel includes specimen type, stain type, orientation of the current brain atlas, dimension of the current overlay view (width $\times$ height $\times$ depth), and actual depth range of the current overlay view. Fig. 5.1 shows the interface of KESMBA v2 containing all the above mentioned features. For more detail, please see the caption for Fig. 5.1.

5.3 Results

In this section, I will present results of the KESMBA v2 framework highlighting the two KESM Golgi data sets. Although both data sets were obtained from whole brains using the KESM, one does not have the left frontal lobe, part of the left temporal lobe, and part of the right frontal lobe due to a misconfigured frame buffer that truncated the images during sectioning and imaging. The other contains the entire brain, but has more noise.

Both data sets are available within the KESMBA v2 framework. The results of the KESMBA v2 framework are shown in Figures 5.2 to 5.5. Besides the whole sections of two Golgi data sets in the horizontal plane, I generated the coronal and the sagittal views for the data sets. Figure 5.3 is the whole sections in the coronal plane and Figure 5.4 the sagittal plane. The original data sets have a voxel resolution of 0.6 μm $\times$ 0.7 μm $\times$ 1.0 μm, so I fixed the voxel ratio to be 1 μm on each axis.

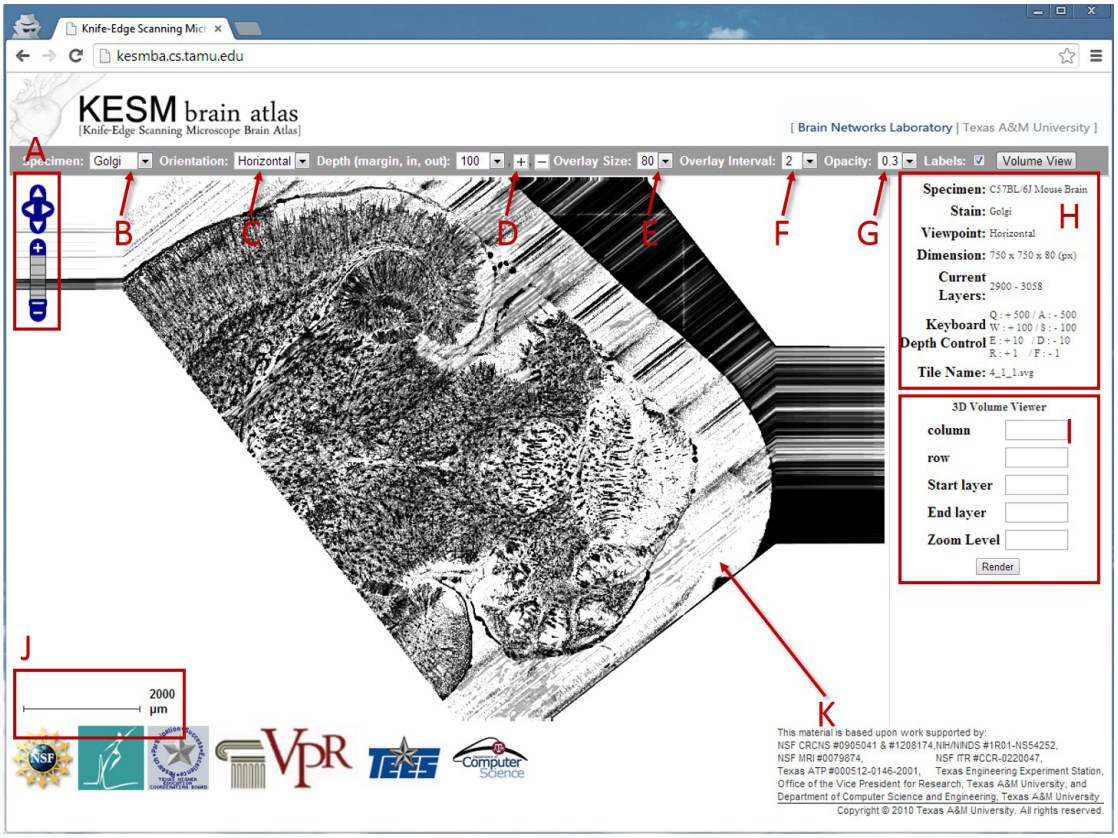


Figure 5.1: KESM Brain Atlas v2 Interface. A screenshot of KESMBA v2 running in a web browser is shown. Red markers and text were added on top for the purpose of explanation, below. A. Navigation panel: panning and zoom-in/zoom-out. B. Data set selection. Golgi, Golgi2, IndiaInk are available in the pull-down menu. C. Sectioning plane orientation. Three standard planes supported (planned). D. Depth navigation. Amount of movement (unit = 1 μm) in the z direction and forward (deeper, [+]) or backward (shallower, [-]) can be controlled. E. Overlay count. How many images to overlay can be selected here. F. Overlay interval. Overlaying evenly spaced at n μm intervals helps visualize thicker sections. G. Opacity. The opacity of layers can be adjusted. H. Information window containing specimen meta data and current location information. I. The input parameters of the volume viewer and the `Render` button to request opening the volume viewer. See Chapter 6 for details. J. Scale bar that automatically adjust to the given zoom level. K. Main display with OpenLayers.

5.3.1 3D Rendering through Image Overlays

The 3D structures in the data are most notable in Figure 5.6. Overlays are effective in rendering 3D content, all within a standard web browser without any dedicated plug-in. To show the effectiveness of the overlaying technique, Figure 5.6A–D show a fixed region by varying the number of overlays. Also, to overcome the limited number of overlays (up to 80), KESMBA v2 can show thicker tissue sections by adjusting the overlay interval size. For example, overlaying 80 images at the interval of 2 would visualize a 160-μm-thick volume, compared to a 80-μm-thick volume at the interval of 1 (Figure 5.6D and E).

5.3.2 Multiscale Nature of the KESM Data

KESMBA v2 supports easy navigation through the data, both within a certain scale and across multiple scales. Figures 5.7–5.8 show the multiscale nature of the KESM data and the effectiveness of KESMBA v2 framework in handling such multiscale data. Figure 5.7 shows successive snapshots of the horizontal plane from KESMBA v2 by gradually zooming into the hippocampus from the largest scale to the smallest scale. Figure 5.8 shows successive snapshots of the sagittal plane from KESMBA v2 by gradually zooming into the cerebellum. Each step of zooming in doubles the resolution, so the final panel has 16$\times$ higher resolution than the first panel.

5.3.3 Transform KESMBA Data to Support All Three Standard Views

KESMBA v1 only supports the native orientation of the data sets. The two Golgi data sets we have were obtained in horizontal sections. However, restricting to a single orientation hampers exploration and understanding of the data. To minimize the need to download unit volumes for full 3D inspection, a good alternative is to

provide all three standard orientations: horizontal, coronal, and sagittal. KESMBA v2 provides all three standard orientations of one of the Golgi data sets: See Figure 5.2 for horizontal, Figure 5.3 for coronal, and Figure 5.4 for sagittal. Figure 5.9 shows details of the hippocampus and the cerebellum from multiple orientations. Whole structures of Purkinje cells can be better appreciated from these three standard orientations. KESMBA v2 has a pull-down menu to choose between these three standard orientations.

5.3.4 Download Performance

I used HttpWatch 9.2 tool (`http://www.httpwatch.com/`) with Firefox 27.0.1 browser to measure the page download time at different overlay numbers in KESMBA v1 and v2 to evaluate their performances. Fig. 5.10 shows two measurements: one is the size of sum of the tiles and the other is the page download time at different overlay numbers. As I expected, the download size of the PNG tiles in KESMBA v1 are more than five times as much as that of the SVG files in KESMBA v2. The decreased tile size in KESMBA v2 caused to highly reduce the page loading time compared with KESMBA v1 (Fig. 5.10B). The reduced file size also allows increasing overlay numbers up to 80 in KESMBA v2. However, the highly decreased size of the tiles in KESMBA v2 does not cause losing structure information.

To estimate the SVG conversion error, I checked the pixel-wise mismatch between the original PNG and the converted SVG tiles in the Golgi data set. For this, I selected 10 layers at an interval of 500 μm along the axial direction. In each layer, I compared all tiles (PNG vs. SVG) at zoom level 0, the highest resolution, and counted the number of mismatched pixels (after thresholding). The average percentage of the mismatched pixels was only 0.7 %, and the standard deviation 0.2 %.

5.4 Summary

KESMBA v2 using OpenLayers allows faster navigation and deeper visualization of KESM data sets compared to KESMBA v1. SVG tiles of KESM data are noticeably decreased in size compared to PNG tiles used in KESMBA v1. Because of the reduced file size, the tiles into KESMBA v2 is not only rendered faster, but also enables overlays of up to 80 layers. The transition from Google Maps API to OpenLayers API allows open-ended customization and maintenance. Both Google Maps and OpenLayers fundamentally catered toward 2D data, so I developed an overlay-based 3D rendering approach. The approach was also very light-weight, only using standard HTML and Javascript to achieve a quick, effective, and resource-efficient web-based 3D visualization of massive volume data. Furthermore, KESMBA v2 supports all three standard orientations (horizontal, coronal, and sagittal) to minimize the need to download unit volumes for full 3D inspection. I expect the KESMBA v2 framework to accelerate neuroscience research.

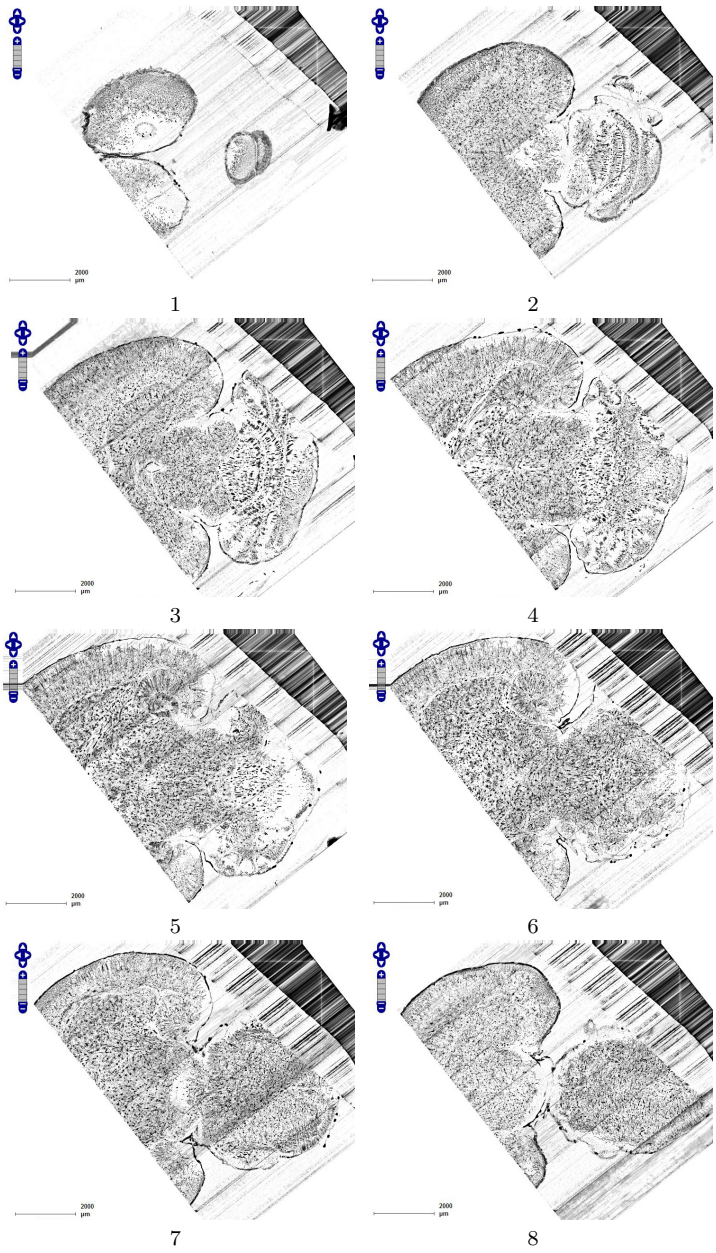


Figure 5.2: Golgi Data Set 1 in the Horizontal Plane. A fly-through of the Golgi data set 1 in the horizontal plane is shown (left: anterior, right: posterior). This is the full extent of the data that was captured. We can see that part of the left temporal lobe, left frontal lobe, and part of the right frontal lobe are cut off which was due to a misconfigured frame buffer during imaging. Scale bar = 2 mm. Each image is an overlay of 80 images in the z direction. The z-interval between each panel is about 700 μm. The numbers below the panels show the ordering. These are cropped screenshots from KESMBA v2.

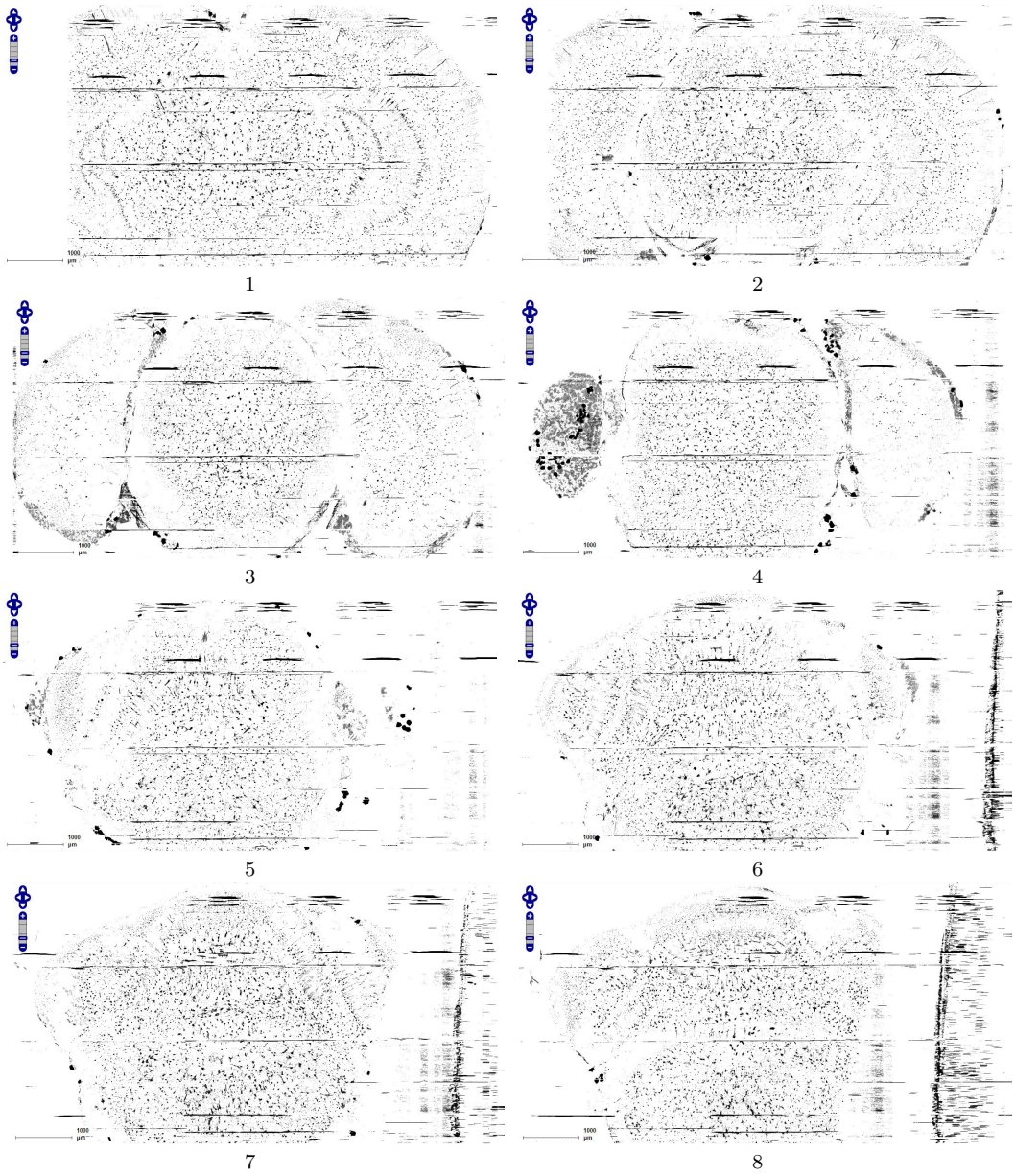


Figure 5.3: Golgi Data Set 1 in the Coronal Plane. A fly-through of the Golgi data set 1 in the coronal plane is shown. Scale bar = 1 mm. Each image is an overlay of 80 images at an interval of 2, representing a 160-μm-thick volume. The z-interval between each panel is about 500 μm. The numbers below the panels show the ordering. These are cropped screenshots from KESMBA v2.

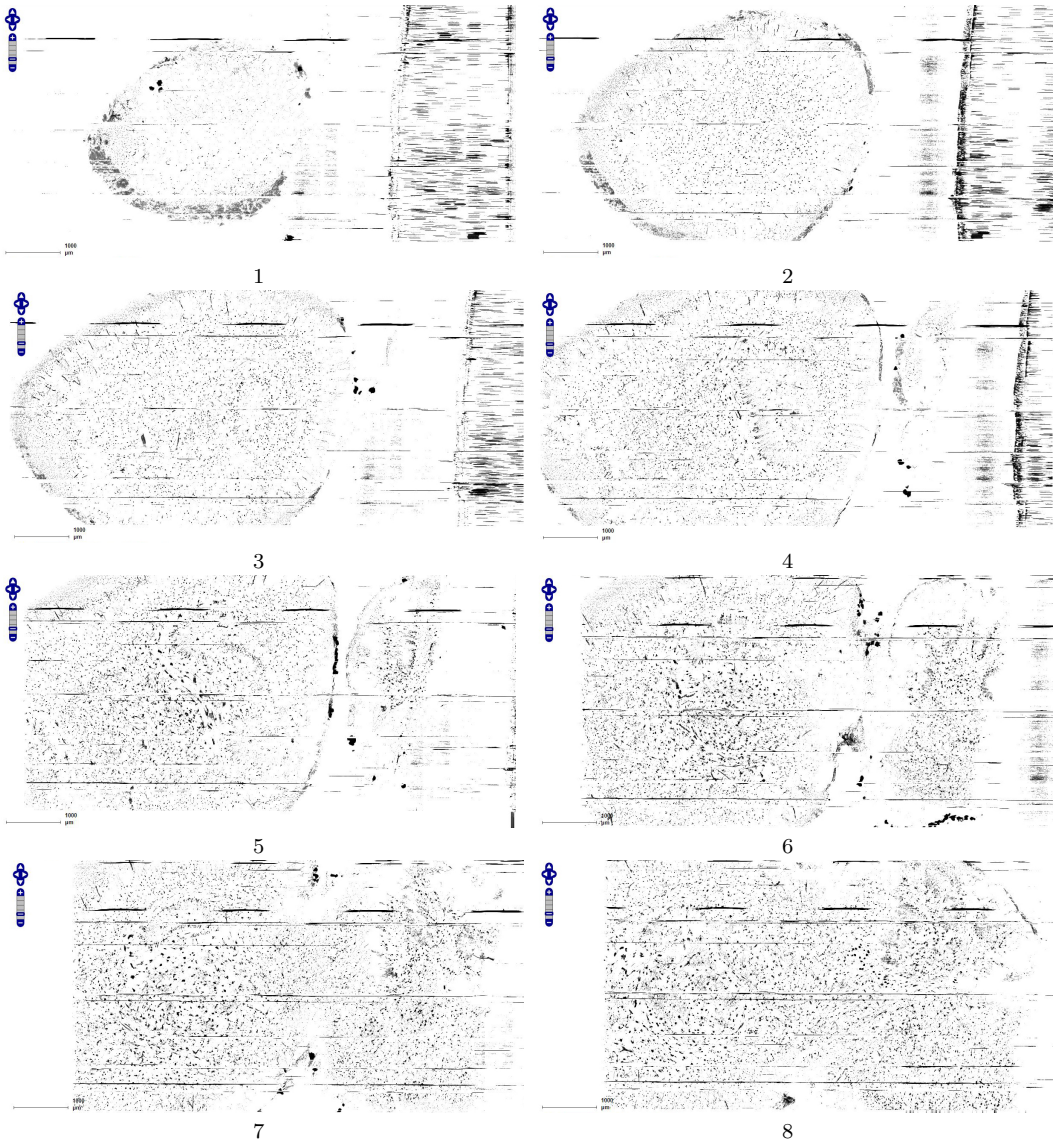


Figure 5.4: Golgi Data Set 1 in the Sagittal Plane. A fly-through of the Golgi data set 1 in the sagittal plane is shown. Scale bar = 1 mm. Each image is an overlay of 80 images at an interval of 2, representing a 160-μm-thick volume. The z-interval between each panel is about 500 μm. The numbers below the panels show the ordering. These are cropped screenshots from KESMBA v2.

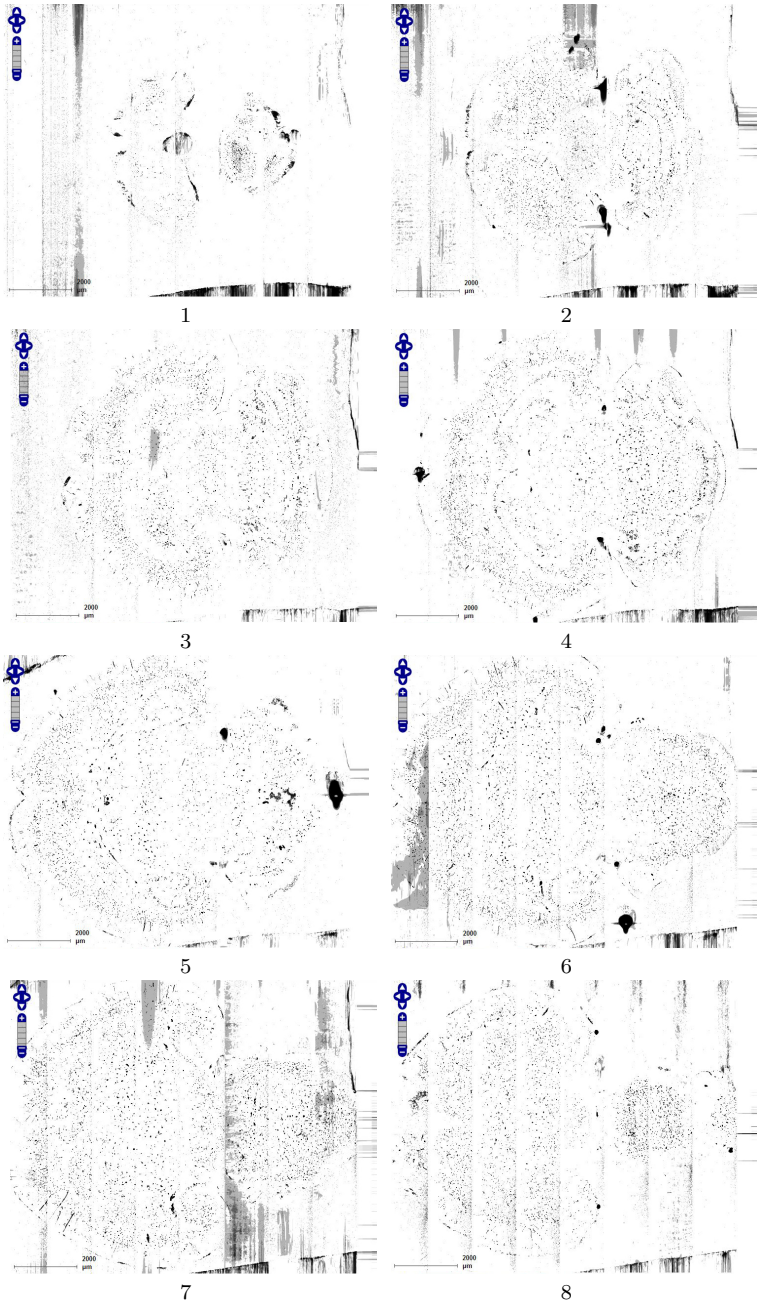
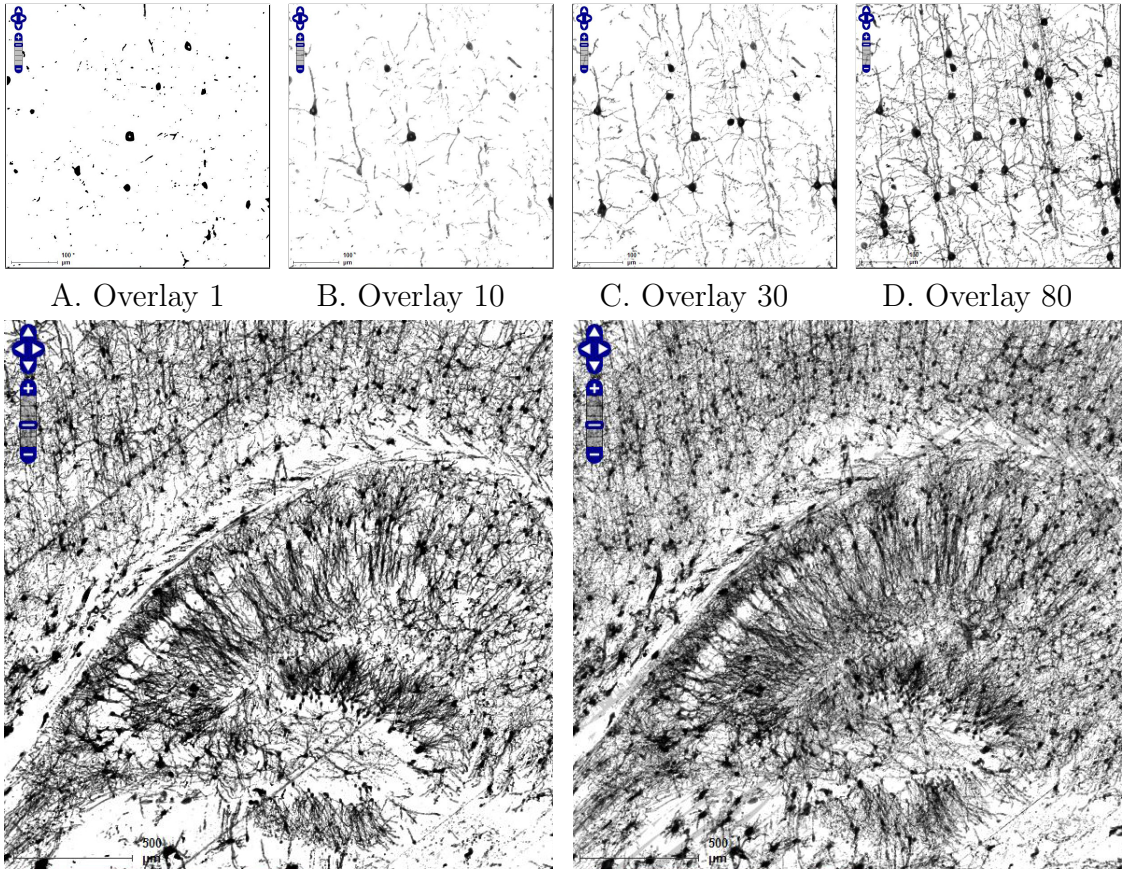


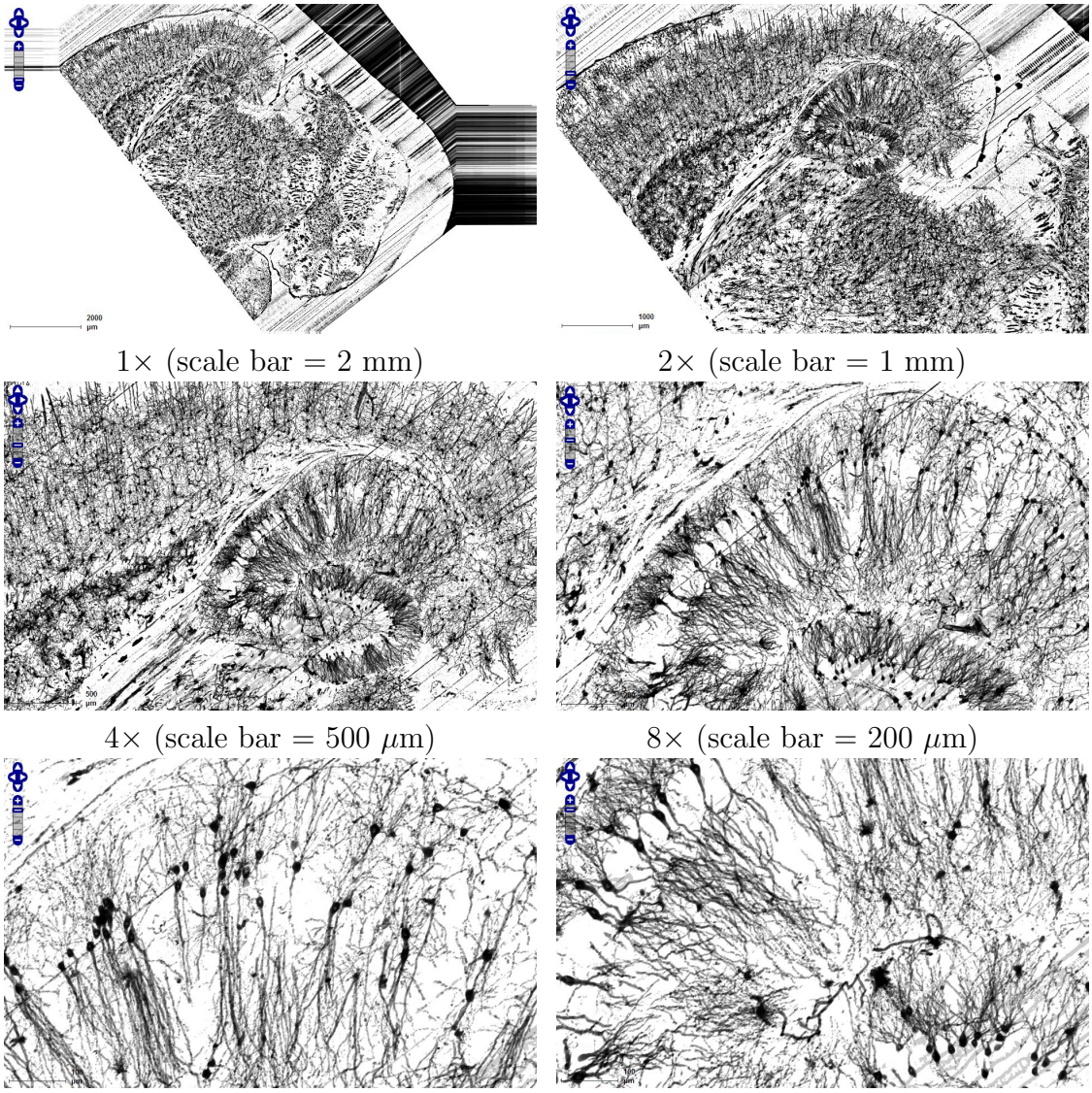
Figure 5.5: Golgi Data Set 2. A fly-through of the Golgi Data Set 2 is shown. The data were obtained by sectioning in the horizontal plane (left: anterior, right: posterior). Scale bar = 2 mm. Each image is an overlay of 80 images in the z direction. The z-interval between each panel is 700 μm. The numbers below the panels show the ordering.



A. Overlay 1 B. Overlay 10 C. Overlay 30 D. Overlay 80

E. Overlay 80 (interval 1) F. Overlay 80 (interval 2)

Figure 5.6: Effectiveness of Image Overlays. A–D. The effect of an increasing number of overlays is shown at the interval of 1. Scale bar = 100 μm. A. Since each KESM image corresponds to a 1-μm-thick section, a single image conveys little information about the neuronal morphology. B. 10 overlayed images, corresponding to a 10-μm-thick section, begin to show some structure but they are not enough. C. With 30 overlayed images, familiar structures begin to appear. D. 80 overlayed images, showing the circuits more clearly. E–F. In E, 80 overlays at an interval of 1, representing a 80-μm-thick volume, are shown. In F, 80 overlays at an interval of 2 are shown, representing 160 μm. Even though D shows the clear dendritic arbor in the hippocampus (left), the massive number of pyramidal cells and their apical dendrites (right) are more densely visible in E.



1× (scale bar = 2 mm)

2× (scale bar = 1 mm)

4× (scale bar = 500 μm)

8× (scale bar = 200 μm)

16× (scale bar = 100 μm)

16× (scale bar = 100 μm)

Figure 5.7: Multiscale View of the Hippocampus in the Horizontal Plane. A multiscale view in the horizontal plane from KESMBA v2 is shown (Golgi data set 1), by gradually zooming into the hippocampus. The numbers below the panels show the zoom-in scale. All panels show an overlay of 80 layers with an interval of 1.

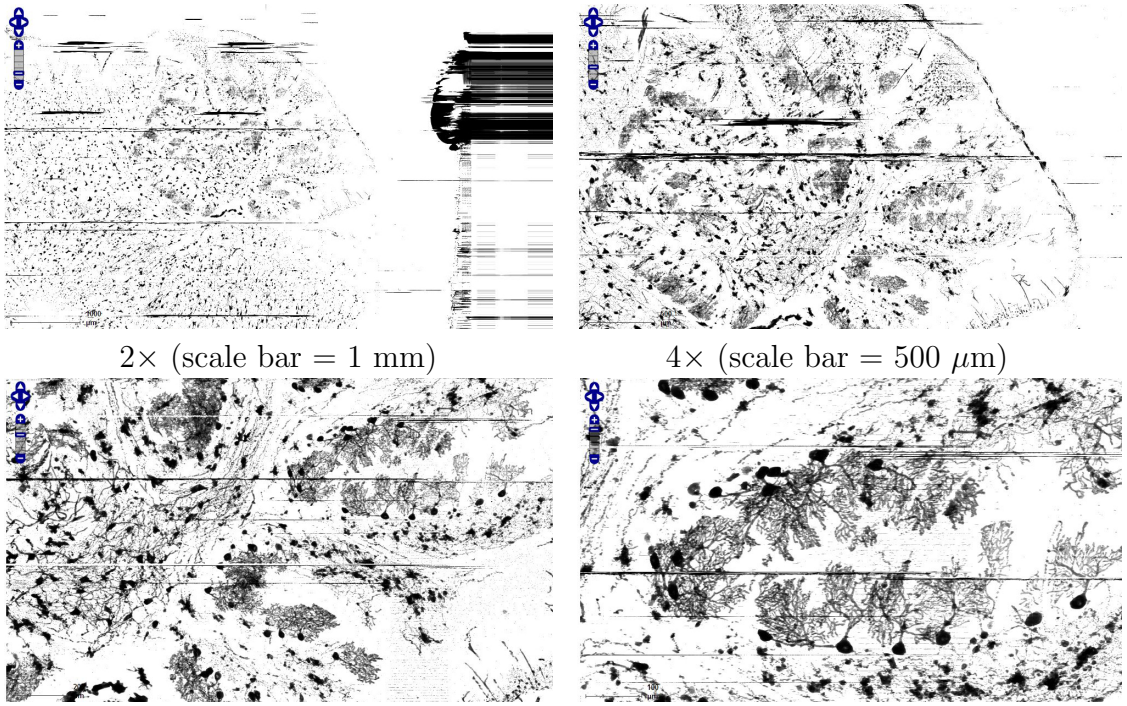
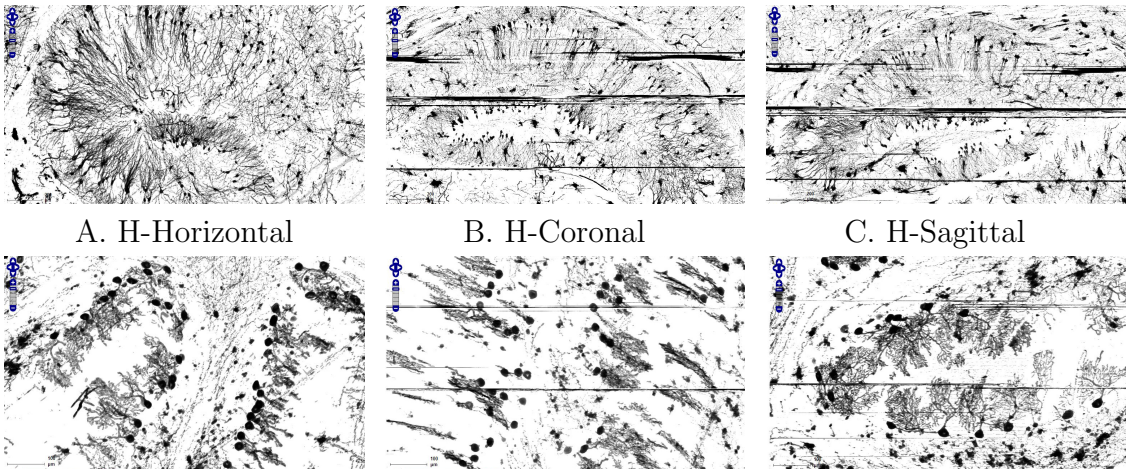


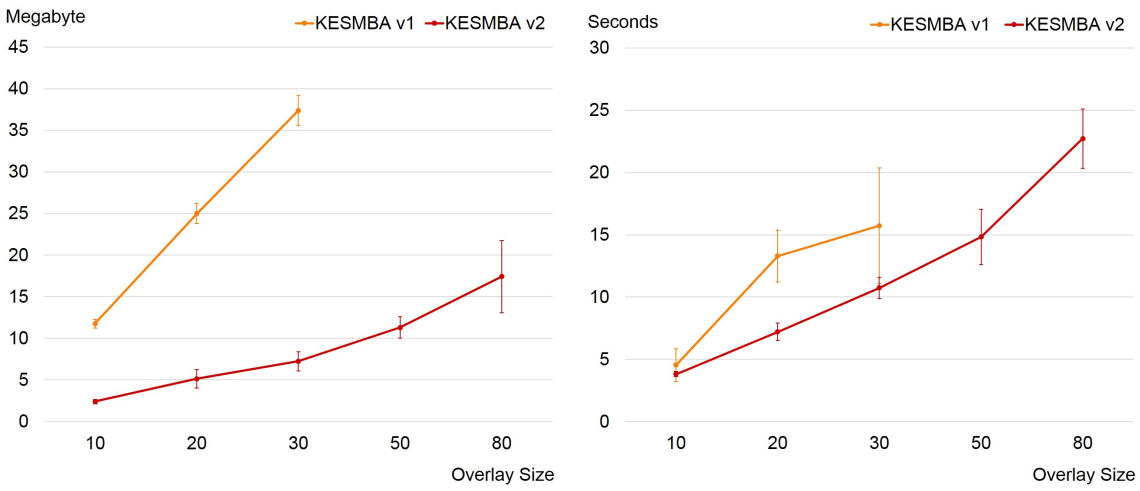
Figure 5.8: Multiscale View of the Cerebellum in the Sagittal Plane. A multiscale view in the sagittal plane from KESMBA v2 is shown (Golgi data set 1), by gradually zooming into the cerebellum. The numbers below the panels show the zoom-in scale. All panels show an overlay of 80 layers with an interval of 2. The panel of the scale 16 shows the structures of Purkinje cells within the Purkinje layer in the cerebellum.



A. H-Horizontal B. H-Coronal C. H-Sagittal

D. C-Horizontal E. C-Coronal F. C-Sagittal

Figure 5.9: Comparison of Different Orientations. Change in the orientation reveals more intuitive details. A–C. The hippocampus, seen from multiple orientations. Overlaying 80 layers with scale bar 200 μm at an interval of 1. D–F. Cerebellar Purkinje cells. In the different orientations, a batch of Purkinje cells located in different orientations were captured. Overlaying 80 layers with scale bar 100 μm at an interval of 2.



A. Data size comparison

B. Download performance comparison

Figure 5.10: Download Performance Comparison Between KESMBA v1 and v2 Data. A. This graph shows the download data size comparison between KESMBA v1 (orange) and KESMBA v2 (red) with different numbers of overlay. B. This graph shows the download performance comparison between KESMBA v1 (orange) and KESMBA v2 (red) with different numbers of overlay. Average of 10 download trials for each setting (browser type and overlay size) is plotted (error bars indicate standard deviation).

6. INTERACTIVE 3D-VOLUME VISUALIZATION AND RECONSTRUCTION OF SMALL UNIT VOLUMES

KESMBA (KESM Brain Atlas) is ideal for surveying large volumes of data, but since the viewpoint is fixed, it can be hard to appreciate the full 3D morphology of the objects embedded in the data set. In order to overcome the limitation of KESMBA and to complement it, I developed a web-based interactive 3D-unit-volume viewer, capable of providing an interactive view and analysis of the volume data. Besides interactive volume visualization, the volume viewer includes a volume reconstruction feature by calculating the centerlines and diameters of the filamentary objects (e.g., vascular segments). Through the web-based 3D volume viewer, neuroscientists can not only have interactive access to the full 3D morphology of the objects, but also perform quantitative analysis of a region of interest in 3D objects real-time.

First, the intensities of voxels in a series of 2D SVG tiles or PNG tiles is generated as an input to the volume viewer at the server side. Next, the centerlines are extracted from the intensities of voxels using a thinning algorithm [71]. Based on the extracted centerlines, connected centerline voxels are grouped to form separate segments. The diameters of the structures are measured using a ray casting approach [100]. Then, the voxels of the input volume and the centerlines, formed segments, and diameters are transferred to the client. At the client side, the input volume using its voxels and its reconstruction results based on the segment results and diameters are visualized. The volume viewer utilizes `Three.js` (`http://threejs.org/`) built on top of WebGL and `dat.gui` (`https://code.google.com/p/dat-gui/`, a graphical user interface JavaScript library). WebGL is natively supported by most modern web browsers.

6.1 Methods

6.1.1 Conversion of a Tile Stack into a Byte Array

Users can choose PNG tiles or SVG tiles as an input. An image stack from the PNG tiles needs to be binarized to segment vessels from the background. Among various thresholding approaches, I used Otsu's method which is an effective thresholding method. An image stack from the SVG tiles is rasterized using the transcoder API provided by the Apache Batik library. I stored the binary results of the PNG tiles or the rasterized results of the SVG tiles into a byte array.

6.1.2 Thinning Process

Accurate segmentation of vascular structures from 3D volumetric data is a critical task for the analysis of vascular morphology. Lesage *et al.* [45] developed an algorithm to trace neurovascular networks from 3D volume data. Another effective approach is the vector tracing method [23, 11, 2, 47]. Frangi *et al.* [23] proposed surface snakes represented by tensor product B-spline surfaces that employ a vessel discriminant filter, which enhances tubular structures and simultaneously decreases other morphological structures while suppressing noise.

Sun [89] introduced an automated algorithm for identifying vessel contours in coronary arteriograms using the spatial continuity property of the vessel centerline points. Wink *et al.* [100] presented an iterative procedure, which tracks the central vessel axis, computing the minimum diameter of the vessels. In order to determine the vessel center, they computed the maximum center likelihood by casting rays from a seed point. A new plane perpendicular to the vessels is computed based on a vector made of the last two points on the current central vessel axis. Aylward *et al.* [6] presented an intensity ridge traversal technique for extracting the centerlines of tubular objects and then included multi-scale heuristics and optimal-scale measures

to make the algorithm robust to the effects of intensity variations due to noise, discontinuities, and singularities in the medical images.

Sofka *et al.* [84] introduced the use of likelihood ratio to measure the "vesselness" for extracting vessels in retinal images. The likelihood ratio measure employed multi-scale matched-filter responses, combining vessel boundary measures. Manniesing *et al.* [50] proposed an evolving surface technique under the skeleton of the intermediate segmentation results for tracking 3D vessel axes in computed tomography angiography (CTA) data.

However, these problem-based approaches have limitations as a universal approach for recent high-resolution biomedical image data because of the intensity variations, artifacts, and singularities in such data.

A skeletonization (thinning) approach diminishes an object maintaining its topology. There are several skeleton algorithms to use a set of data points in a 3D volume data or surface patches as an input. To use these algorithms for our data, we need to generate surface patches from our data. I adapted a 3D curve-thinning algorithm [71], which does not require such surface patches. This thinning algorithm extracts geometrically and topologically preserved skeletons.

Palagyi *et al.*'s algorithm [71] first classifies each voxels as border points, line-end points, and simple points based on their local neighbor voxels (the 26-neighbors, the $3 \times 3 \times 3$ neighborhood of each voxels). A simple point is a point whose removal does not alter the topology of the object. A point is a border point if it is a simple point and not a line-end point. Each sequential object reduction process consists of six sub-iterations based on the six surfaces (up, down, left, right, front, and back) in 3D. This algorithm repeats the sequential process until there is no remaining border points.

The detailed thinning steps are as follows:

1. Create a hash map of voxels in the volume, using a `HashMap<Voxel>` data structure.

2. For each voxel in the hash map, check if the voxel satisfies the simple point condition. To check the simple point condition, instead of checking if each voxel's 26 neighbors are in the condition, I used a pre-calculated lookup table (2^{26}) having all possible configurations of 26 neighbors weather each configuration is in the simple point condition or not.

3. If the voxel is a simple point and not a line-end point, I removed the voxel in the map then checked whether its neighbor voxels become surface voxels. If the neighbors are surface voxels, they are added to the map.

4. Repeat steps 2–3 until there are no more simple points.

6.1.3 Labeling Centerlines and Calculating Diameters of Vessels

Based on the extracted centerline of the vessels, separate vessel segments can be labeled by searching adjacent voxels in the centerlines using the Depth First Search (DFS) algorithm. First, a voxel is randomly selected and its neighbors on the centerlines are traced until there is no more adjacent neighboring voxels of the initial voxel. To label the other vessel segments, an unvisited voxel is randomly selected again and the process is repeated. This process repeats until there is no more unvisited voxels on the centerlines. All voxels on a connected vessel are given the same labels. Based on the lengths of each labels, unwanted short side branches are pruned.

Furthermore, the diameters of vessels are calculated for reconstructing the geometrical vessels. First, 26 projections are generated at each voxels on the centerlines

and they are sent towards the boundary of the vessels. 26 projections can be evenly scattered in 3D for not causing any angular bias although the angles are not uniformly spaced. Figure 6.1 shows an example of the 26 projections radiating from a voxel on the centerline. The black cylinder is a vessel and the spheres are voxels of centerlines. Among the spheres the red sphere is the point where the diameter is to be estimated and the 26 projections are generated from this voxel. The 26 generated projections are sent towards the boundary of vessel. Once they meet the boundary, they stop. Based on these projections, I calculated the shortest distance as the diameter of the vessel at that center voxel. All voxels of the centerlines are visited and the diameters calculated in the same manner.

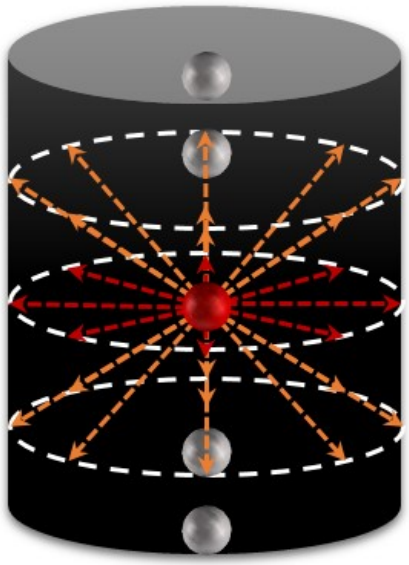


Figure 6.1: Sending 26 Projections from a Selected Voxel. Calculating the diameter of the selected voxel. In the vessel (a black cylinder volume), among the voxels (spheres) of the centerline, a voxel (a red sphere) was selected. The selected voxel has 26 projections (arrows) to calculate its diameter based on the vessel boundary. The red arrows indicate the projections on the XY plane at the selected voxel and the orange arrows the projections towards the up and down planes at the selected voxel.

6.1.4 Data Compression

To transfer the reconstruction results to the client, the visualization framework converts the voxels of the volume and centerlines with labels and diameters to Java Script Object Notation (JSON) format, which is a lightweight plain-text data structure. Google Gson was used for the actual implementation. I compressed the voxels by converting the voxels to an array of pairs, because the vascular network volume data consists of a large number of consecutive zeros or ones. While a vascular volume of $256 \times 256 \times 400$ pixels generates about 56 MB JSON file from an array of voxels, an equivalent JSON file is only 8 MB, a factor of 7 reduction. The reduced size of the file decreases the data transmission time across the network.

6.1.5 Data Transmission and Visualization

For data transmission, I used the Representational State Transfer (REST) web service that retrieves JSON data from the server statelessly. The client uses HTTP GET (a data-producing method) to request JSON data using the following parameters: column number, row number, starting layer number, ending layer number, and zoom level. Once the server generates JSON data, the client retrieves the JSON data using asynchronous JavaScript and XML (AJAX).

JSON data contains two data groups: 1) the sequences of zeros and ones representing the voxels of the data volume and 2) the voxels of the centerlines and the segment labels and diameters corresponding to each voxel. To visualize the JSON data, I used `Three.js` built on top of WebGL.

First, I added all voxels of the volume into the `vertices` in `THREE.Geometry` object. Then, I used the `THREE.ParticleSystemMaterial` object to define the geometry object properties. The properties included color, transparency, and size. To bind these two objects together, I created a particle system using `THREE.ParticleSystem`

object, and then I added the instance of the `THREE.ParticleSystem` object into the scene, the `THREE.Scene` object. For all voxels of the centerlines and their labels and diameters, I used the same manner to visualize them.

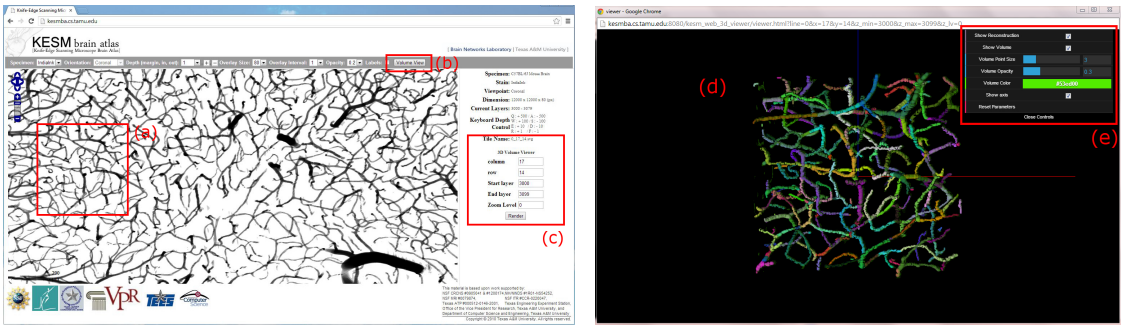
The volume viewer provides multiple options such as the color, the point size, and transparency using `dat.gui` which is a lightweight graphical user interface library.

6.2 Results

In this section, I will present the results of the web-based volume viewer with the KESM Golgi and India ink data sets. Our web server hosts the SVG tiles and PNG tiles of entire Golgi and India ink data sets. The web-based volume viewer provides both SVG and PNG tiles as an input. From an image stack, the proposed viewer visualizes the volume or the geometric reconstruction of the input volume, displaying the reconstruction result.

Figure 6.2 shows a screenshot of KESMBA v2 interface and a screenshot of the web-based volume viewer interface. In order to open the volume viewer, a user needs to input the location and image stack range information in the KESMBA v2 browser window. As the user clicks a tile on the view panel of KESMBA v2, the information panel shows the location and the zoom level of the selected tile. Once the user clicks the `Volume View` button, all input boxes which are required for the volume viewer are displayed. After the user finishes typing in the parameters and clicking the `Render` button, the volume viewer transfers the parameters to the KESMBA web server and requests the centerlines of the volume and their labels and diameters. Based on the results, the volume viewer on the client side shows the input volume and the reconstruction results.

In the Figure 6.3, the procedures of the reconstruction are shown. In order to geometrically reconstruct a given volume, first, we need to extract the centerlines of



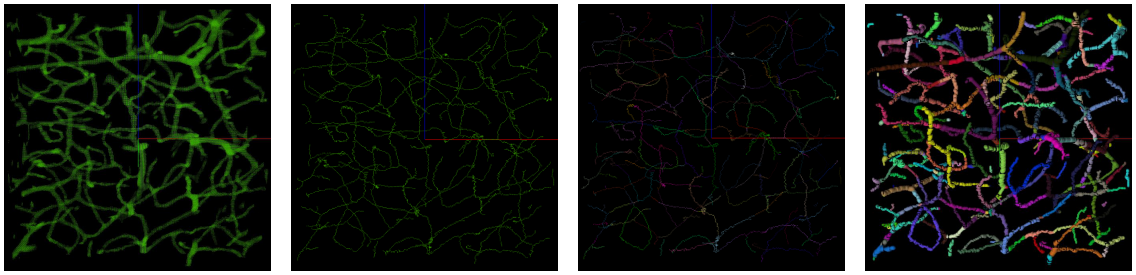
A. KESMBA v2 Interface B. The Volume Viewer

Figure 6.2: The Web-based Volume Viewer Interface. The screenshots of KESMBA v2 and the volume viewer are shown. Red markers and text were added on top for the purpose of explanation. A. The KESMBA v2 interface: (a) A selected tile (location) to render a 3D volume. (b) The volume View button to activate all text boxes related to input parameters of the volume viewer. (c) All text boxes of the input parameters of the volume viewer and the `Render` button to request opening the volume viewer. The volume viewer needs five input parameters – the location information (column number, row number, and zoom level number) of the selected tile and the stack information (start and end layers). B. The volume viewer interface: (d) The view panel. (e) The control panel. The panel consists of a check box for showing reconstruction results, a check box for showing the selected volume, a scale bar to adjust the size of the volume voxels, a scale bar to adjust the volume opacity, a color table to select the volume color, and a check box for overlaying the coordinate axes.

the volume (Figure 6.3B). Then, separate vessels are labeled by searching through unvisited center voxels using DFS (Figure 6.3C). Finally, the calculated diameters of the vessels at each center voxels using a projection method are used to reconstruct the volume (Figure 6.3D).

Figure 6.4 shows the comparison between PNG and SVG volumes from the Golgi and India ink data. The volume views from the PNG image stack provide a slightly better view compared to the SVG image stack, due to minor loss in information due to the bitmap to vector to bitmap conversion.

Figures 6.5–6.6 show comparisons between the 3D views from KESMBA v2 and



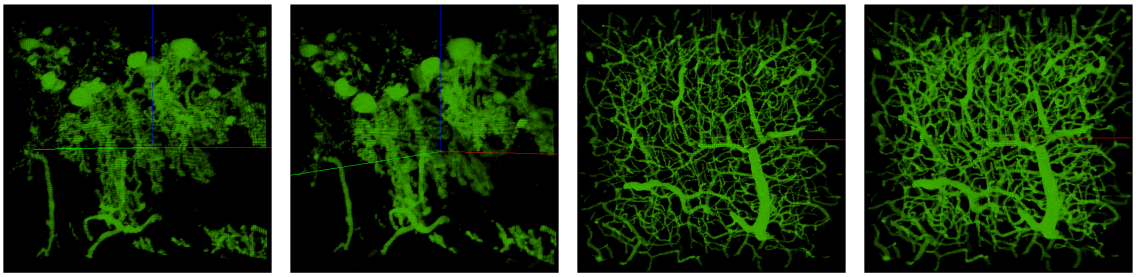
A. India ink Vol. B. Centerlines C. Labeling D. Reconstruction

Figure 6.3: The Reconstruction Results from a Volume. A. A screenshot of the volume viewer showing an India ink volume of 100-image stack. B. The centerline result from the India ink volume (A). C. The labeled result from the India ink volume (A) based on the centerline result (B). D. The reconstruction result from the India ink volume (A), adding the vessel diameters of each voxels based on the labeled centerline result (C).

the volume views of the same regions in the volume viewer. While KESMBA v2 enables the overlay of only 80 image layers, the volume viewer provides stacking of over 200 images as well as different angles of view by interactively rotating the volume.

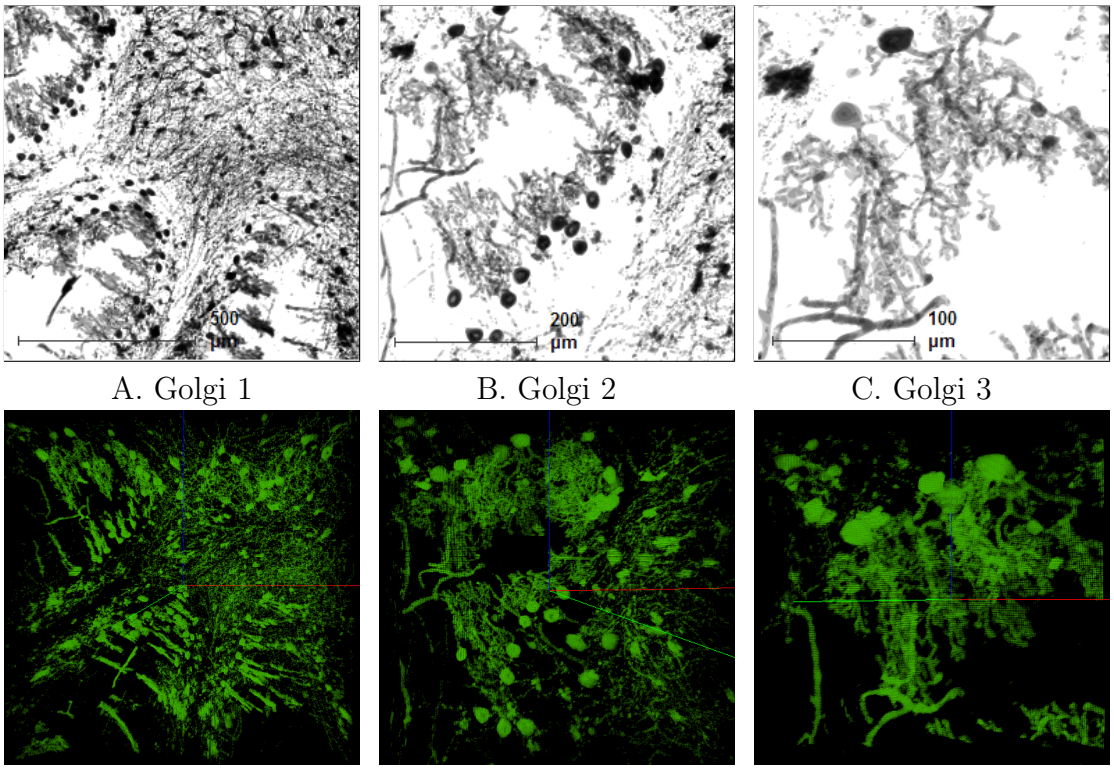
6.3 Summary

In this section, I introduced a web-based volume viewer capable of interactively visualizing unit volumes as well as geometrically reconstructing the unit volumes. The visualization of the volumes is a function that complements the overlay-based rendering of KESMBA v2. The volume viewer also provides the following features for enhancing the view: (1) adjust voxel size, (2) adjust opacity, and (3) change color. On top of these features, it provides the geometrical reconstruction feature from a given volume and visualize the reconstruction results.



A. Golgi PNG B. Golgi SVG C. India Ink PNG D. India Ink SVG

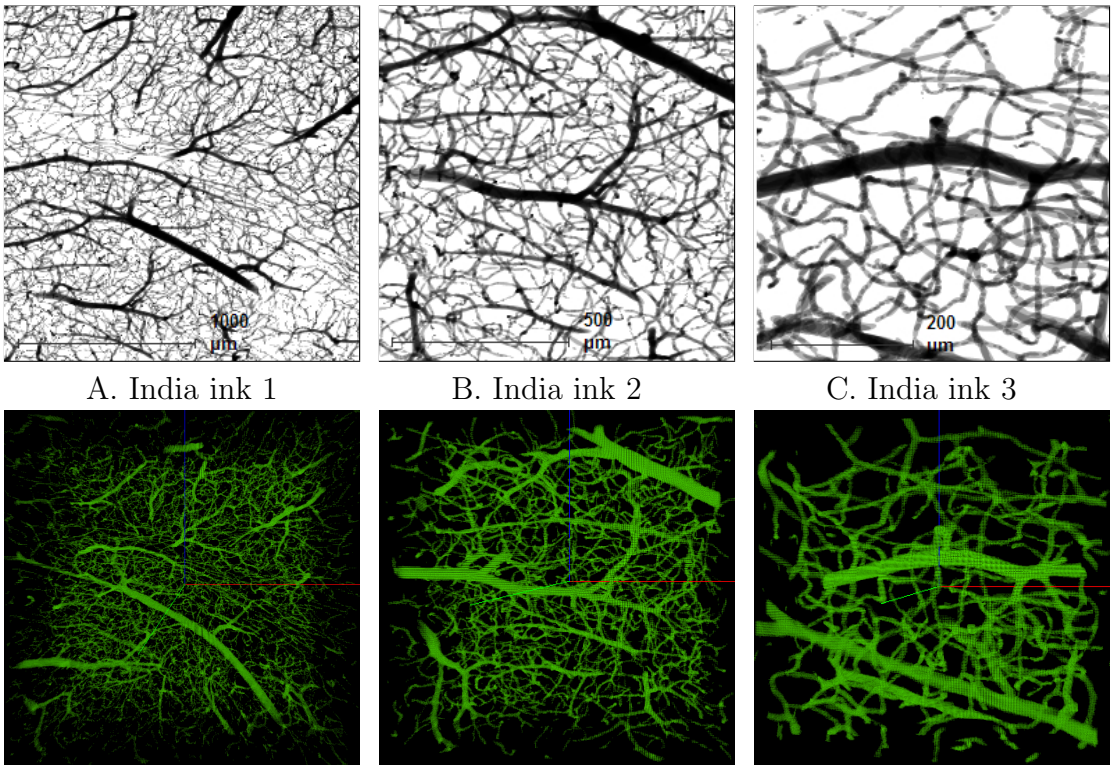
Figure 6.4: A Comparison Between PNG and SVG Volumes. A. 100-PNG-image stack of the Purkinje cells in the cerebellum from the Golgi data. B. 100-SVG-image stack of the same region with A. C. 100-PNG-image stack in the hippocampus area from the India ink data. D. 100-SVG-image stack of the same region with C.



A. Golgi 1 B. Golgi 2 C. Golgi 3

D. Golgi 1 Volume E. Golgi 2 Volume F. Golgi 3 Volume

Figure 6.5: A Comparison Between KESMBA v2 and the Volume Viewer in the Golgi. A–C. The structural views at the different zoom levels in the cerebellum of the Golgi data from KESMBA v2, using 80 overlays and 2 μm interval between overlays. A. A structural view of the cerebellum at two-time zooming-out level (4 times reduced), the scale bar 500 μm. B. A structural view of the cerebellum at one-time zooming-out level (2 times reduced), the scale bar 200 μm. C. A structural view of the cerebellum in the original size, the scale bar 100 μm. D–F. The volume views at the different zoom levels in the cerebellum of the Golgi data from the volume viewer, using 100-image stack. D. A volume view of the same region with A. E. A volume view of the same region with B. F. A volume view of the same region with C.



A. India ink 1 B. India ink 2 C. India ink 3

D. India ink 1 Volume E. India ink 2 Volume F. India ink 3 Volume

Figure 6.6: A Comparison Between KESMBA v2 and the Volume Viewer in the India Ink. A–C. The structural views at the different zoom levels in the hippocampus of the India ink data from KESMBA v2, using 80 overlays and 2 μm interval between overlays. A. A structural view of the hippocampus at two-time zooming-out level (4 times reduced), the scale bar 500 μm. B. A structural view of the hippocampus at one-time zooming-out level (2 times reduced), the scale bar 200 μm. C. A structural view of the hippocampus in the original size, the scale bar 100 μm. D–F. The volume views at the different zoom levels in the hippocampus of the India ink data from the volume viewer, using 100-image stack. D. A volume view of the same region with A. E. A volume view of the same region with B. F. A volume view of the same region with C.

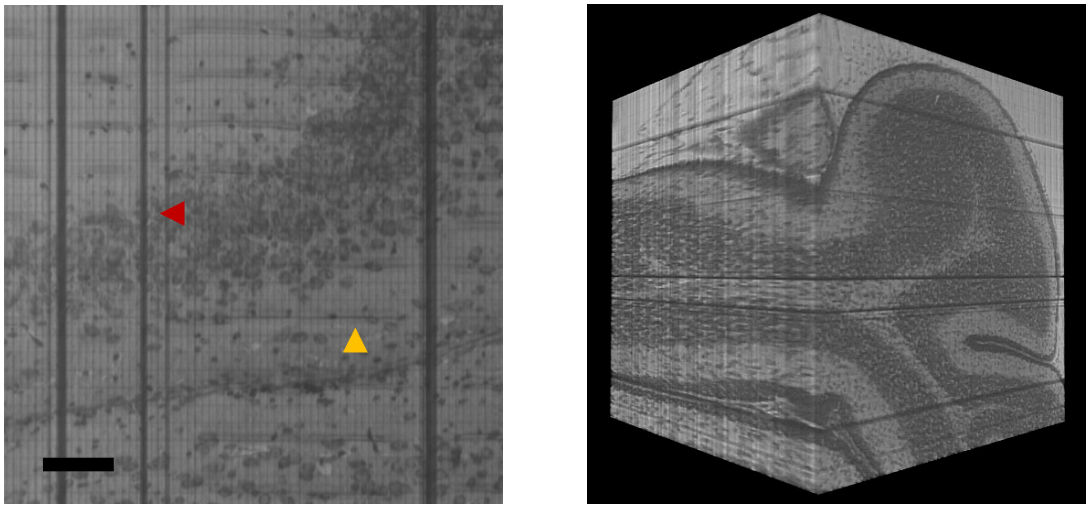
7. MAPPING KESM DATA TO THE WAXHOLM SPACE

In this section, I address two major challenges: (1) coping with the unique source of noise in the KESM data sets, and (2) increasing the usefulness of our high-resolution, high-quality data (Nissl-stained, showing neuronal cell bodies). Our approach is to (1) use a localized- and global-FFT-based denoising algorithm to remove noise that is characteristic of the KESM modality, and (2) registering our mouse brain data (with a landmark-based rigid transformation method using Moving Least Squares) to a standard atlasing space (Waxholm space, a new standard coordinate space for rodents [30]) so that we can import valuable annotations (such as boundary of anatomical substructures or gene expression data) from existing mouse brain atlases.

7.1 Image Noise Removal

A variety of noise removal techniques exists for medical images such as wavelet-based [101, 22, 9], ridgelet- and curvelet-based [88], and Fourier-wavelet-based [75] approaches.

However, KESM has unique noise characteristics due to its use of physical sectioning coupled with simultaneous imaging, so a customized noise removal method is necessary. For example, tiny defects at the knife edge cause streaks. Also, the cutting process induces irregular marks due to vibration of the knife, which is called "chatter". Fig. 7.1 shows the typical noise patterns observed in KESM images. To address these issues, previously we developed an approach based on normalization and localized mean filter [58], but the approach was not able to scale up to whole brain data sets.



(a) Raw Nissl image (b) Raw Nissl volume

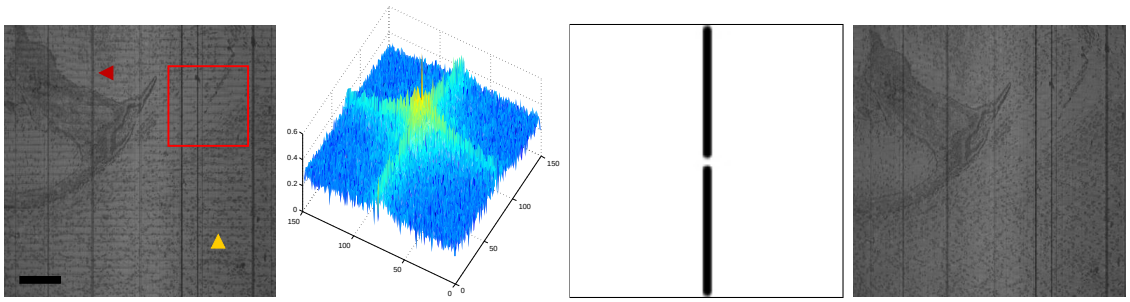
Figure 7.1: Raw Nissl Data. (a) Single image. The red arrow indicates a vertical streak and the yellow arrow knife chatter. Scale bar = 47 μm. (b) A 1.05 mm cube (cerebellum).

7.1.1 Methods

For denoising, sequentially apply (1) localized frequency domain filtering to remove noise due to knife chatter and (2) global frequency domain filtering to remove streaks due to defects on the knife edge.

Localized FFT: First, in order to process the knife chatter noise as a periodic noise pattern, we subdivide the original image into small subimages where the chatter marks show a regular pattern in the frequency domain. We apply the notch reject filter to all subimages to remove the knife chatter noise (cf. the noise removal technique used in [46]). Fig. 7.2 shows the localized frequency domain filtering process and the results of the process.

The procedure we followed was as follows. First, from the image $I(x, y)$ of size



(a) (b) (c) (d)

Figure 7.2: Removal of Knife Chatter Noise. (a) From a cropped original image in the lateral septal nucleus, we determined a small window size so that the small window (the red square) could contain periodic chatter patterns that could be easily detected in the frequency domain. The red arrow indicates a vertical streak and the yellow arrow knife chatter, scale bar = 144 μm. (b) The Fourier spectrum from the red square region in (a). The knife chatter noise can be seen as peaks along the y-axis. (c) A notch reject filter for denoising, applied to (b). (d) All knife chatters disappeared from the original image after applying the notch reject filter to all subimages.

$M \times N$, its 2D FFT $F(u, v)$ is calculated:

$$F(u,v) = \frac{1}{MN} \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} I(x,y) e^{-2\pi i(\frac{ux}{M} + \frac{vy}{N})},$$

where $u = 0, 1, ..., M-1$ and $v = 0, 1, ..., N-1$. Next, we shift the zero frequency component to the center of F:

$$F_{\mathrm{s}}(u,v) = F(u - \frac{M}{2}, v - \frac{N}{2}).$$

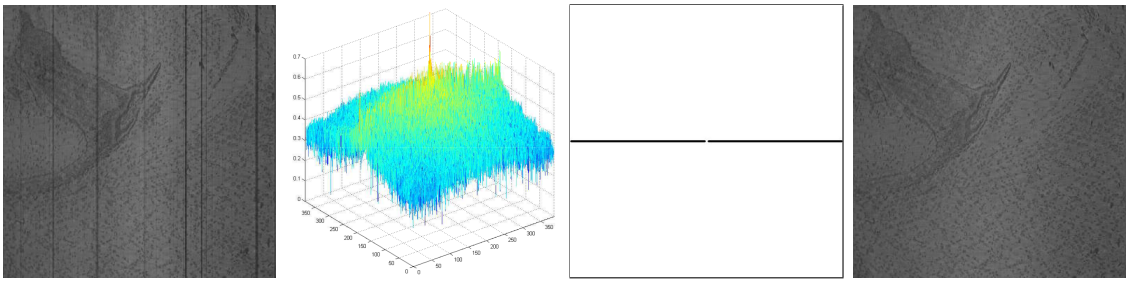
Then, using a notch reject filter $B(u, v)$, Fig. 7.2(c), block out the zero frequency component in F_{s}:

$$F_{\mathrm{s,d}}(u,v) = F_{\mathrm{s}}(u,v) * B(x,y).$$

Note that we are convolving ("$*$") the shifted Fourier spectrum F_{s} with B to denoise F_{s}. Finally, shift back the zero frequency component of $F_{\mathrm{s,d}}(u, v)$ to the original

position and inverse-FFT the result:

$$\begin{aligned} F_{\mathrm{d}}(u,v) &= F_{\mathrm{s,d}}(u-\frac{M}{2}, v-\frac{N}{2}) \\ I_{\mathrm{d}}(x,y) &= \sum_{u=0}^{M-1}\sum_{v=0}^{N-1} F_{\mathrm{d}}(u,v)e^{2\pi i(\frac{ux}{M}+\frac{vy}{N})} \end{aligned}$$



(a) (b) (c) (d)

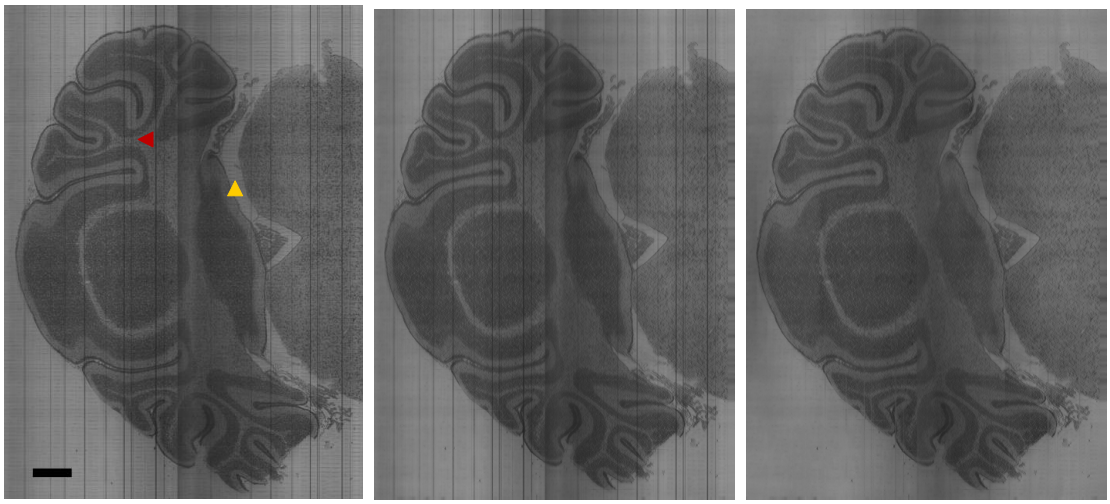
Figure 7.3: Removal of the Streak Artifacts. (a) The knife chatter was suppressed, but streak artifacts remained. (b) The Fourier spectrum of (a). (c) A notch reject filter for denoising. (d) The final result of notch reject filtering is displayed.

Global FFT: Next, in order to remove the remaining streak artifacts due to defects in the knife edge, we adopt a global frequency domain filtering approach with a notch reject filter.

First, we apply FFT on the input image $I(x,y)$. We can see the streak artifacts show up along the horizontal axis. Next, we generate a notch reject filter to suppress the region where the periodic streak artifacts are located in the Fourier spectrum. After multiplication of the FFT coefficients with the notch reject filter, we finally acquire the denoised image via inverse Fourier transform. Fig. 7.3 shows the global FFT procedure step by step.

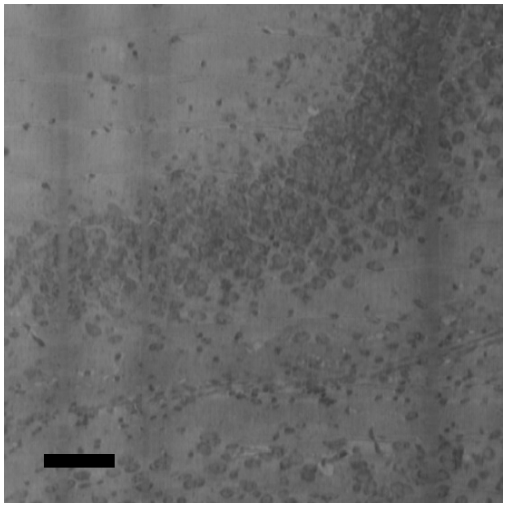
7.1.2 Denoising Results

Fig. 7.4 shows the localized FFT and the global FFT denoising results. When the localized FFT results from neighboring windows were stitched together, artifacts could appear at the boundary. To avoid this artifact, we used slightly overlapping windows before applying the localized FFT filter, and cropped off the boundary (10 pixels) from the results. We also tried the combined wavelet-FFT filtering approach [67] instead of the global FFT and the results were almost identical. Fig. 7.5 is a denoised version of the raw data shown earlier in the Fig. 7.1. In all cases, we can see that the periodic noise is effectively removed.

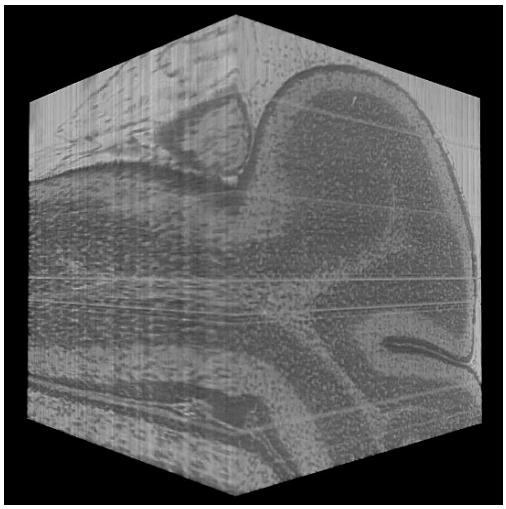


(a) Raw image (b) Localized FFT (c) Global FFT

Figure 7.4: Denoising in Two Steps Using Localized- and Global-FFT. (a) A cropped raw image from the cerebellum. The red arrow indicates a vertical streak and the yellow arrow knife chatter. Scale bar = 300 μm. (b) Localized FFT filtered result. We used 150 $\times$ 150 window size, where the full image itself was 4602 $\times$ 6334. (c) The final result after applying the global FFT filtering.



(a) Denoised result of the Nissl



(b) Denoised result of the Nissl volume

Figure 7.5: Denoised Nissl Data. Denoised version of data in Fig. 7.1. See Fig. 7.1 caption for details. Bright horizontal bands are missing data.

7.2 Registration to the Waxholm Space

Standard coordinate spaces for brain atlases have served a very important role in brain research. A standard coordinate space exists for the human brain, the Talairach space, [91]. In case of the mouse, Franklin et al. [24] provides detailed stereotaxic coordinates. However, a unifying standard has been lacking for the mouse brain. The International Neuroinformatics Coordinating Facility (INCF) developed a canonical coordinate system for the mouse brain called the Waxholm space (WHS) to address this gap [30]. For our work, we used the WHS target volumes reported in [37].

Mutual information-based approaches have been used successfully to register multimodal medical image data [73]. However, the approach does not work well when the image intensity variance is high, and overlooks structural information in the images. Due to these issues, the approach is not suitable for raw KESM data. To preserve local structural properties, a "rigid-as-possible" deformation technique is

needed so that the mapping minimizes distortion (relative shape is maintained). A "rigid-as-possible" deformation method, initially introduced in [3], was simplified by [34], triangulating the source image. However, triangle meshes tend to have poor smoothness.

Based on the above considerations, we adopted a landmark-based 3D registration framework that includes a stable 2D rigid deformation method using Moving Least Squares (MLS) [80]. Initial registration is done through affine transformation (translation, scaling, rotation), and fine-grained registration is done using the MLS-based approach.

7.2.1 Methods

Rigid MLS deformation applies local deformations that satisfy global constraints (user-defined control points) [80].

The formulation below closely follows [80]. Given a set of control points p and their deformed positions q, an optimal deformation function $l_v(\cdot)$ (a rigid transformation mapping p to q) is estimated. Note that the function is defined at every pixel v in the source image. Given a point v in the KESM image, functions $l_v(p)$ that minimizes the following is found:

$$\sum_i w_i \left| l_v(p_i) - q_i \right|^2,$$

where p_i and q_i are vectors and the weights w_i have the form $w_i = \frac{1}{|p_i - v|^{2\alpha}}$ ($|\cdot|$ is the Euclidean distance).

The weight w_i is dependent on the evaluation point v, thus, this method is called "Moving Least Squares minimization". See [80] for a full derivation of the solution to this optimization problem. We first designated corresponding anatomical landmarks between the two data sets as control points/deformation points. The landmarks in the source data set will then be interpolated onto the corresponding landmarks in

the target data set. Based on the corresponding anatomical landmarks, we estimate the deformation parameters between the source and the target images. Once the parameters are determined, we can reuse them in the deformations of all subsequent source images because the images share the same morphological properties.

We aligned the denoised KESM Nissl data set (Sec. 7.1), to the Nissl-stained optical histology atlas in the WHS (`http://software.incf.org/`). Our landmark-based 3D registration framework takes the following steps:

1. Obtain coronal slices from the KESM mouse brain Nissl data.
2. The image stack is scaled, rotated, and shifted in order to coarsely match the coronal slices from the WHS. We scale down the KESM data by a factor of 20 in each dimension.
3. Next, we identify anatomically corresponding locations between the KESM data and the WHS atlas. We use these locations as control points/deformation points.
4. In the deformation step, the corresponding locations between the KESM and WHS data are completely interpolated because 2D rigid deformations using MLS provide the interpolation property.
5. Repeat steps 2-4 with horizontal slices.

7.2.2 Registration Results: 2D and 3D

Fig. 7.6 (a)-(b) show the anatomical landmarks in the KESM atlas and their corresponding locations in the WHS atlas (coronal view). First, based on the images and their landmarks, we estimated the parameters for 2D rigid MLS deformations. With those parameters, we ran the 2D rigid MLS deformations algorithm on the entire set of KESM images (coronal slices) to register them to the WHS. Fig. 7.6 (c) shows a deformed KESM image. Next, we aggregated the deformed coronal KESM

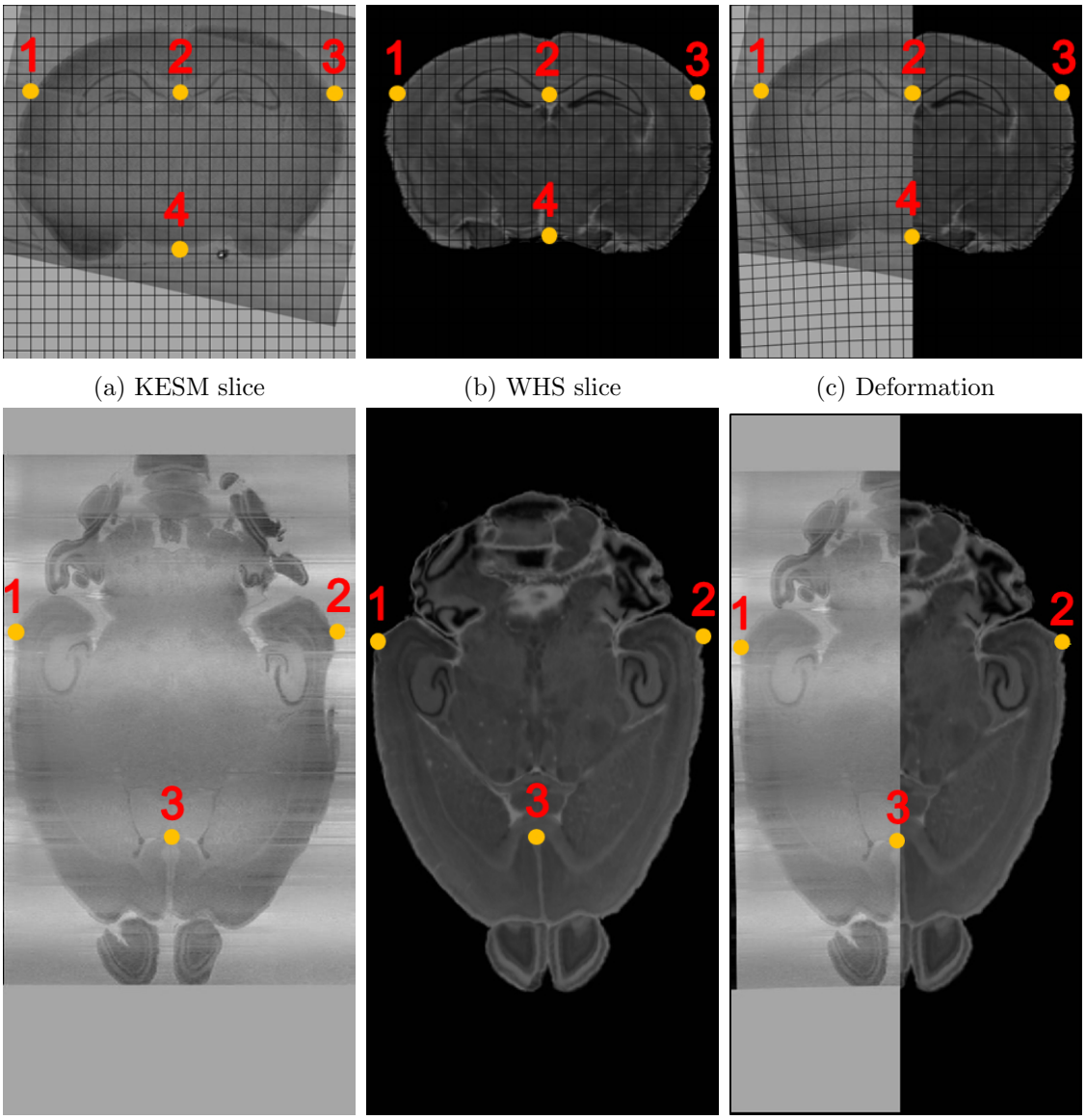
images into a 3D volume, and resliced the volume data set into horizontal slices. Before we deformed these horizontal slices, we removed illumination irregularity along in the planar direction using our FFT-based denoising algorithm because intensity of each coronal slice varied. Finally, we applied the same deformation procedure to acquire the final results. Fig. 7.6 (d)-(e) show the same information as Fig. 7.6 (a)-(b) for the horizontal view. Fig. 7.6 (f) shows an example of deformed KESM image (horizontal view).

Once we deformed all KESM horizontal slices, we reconstructed a 3D volume from the slices. Fig. 7.7 shows a comparison of denoised 2D KESM data registered to the WHS and the corresponding 2D WHS data in the coronal plane. In Fig. 7.7 (a1)-(a3) and (b1)-(b3), the folds of the cerebellum is most visible. In Fig. 7.7 (a4) and (b4), the cortices of the two hemispheres are shown as flanking the brainstem structures in the middle. In Fig. 7.7 (a5)-(a6) and (b5)-(b6), dark folds of the hippocampus can be seen in both hemispheres. In Fig. 7.7 (a7) and (b7), the round mass embedded near the center of each hemisphere is the striatum. Finally, in Fig. 7.7 (a8) and (b8), the olfactory bulb can be easily seen (the two structures near the bottom with ringed layers). KESM has a much higher resolution than the WHS standard atlas, so detailed cellular structure can be observed.

Furthermore, Fig. 7.8 shows comparisons of the (a&d) raw KESM data, (b&e) the denoised and registered KESM data, and (c&f) the WHS data. We can see that the global properties (e.g., aspect ratio) are well aligned. Due to the high-resolution of the KESM data, fine details such as the complex convolutions in the cerebellar cortex are also visible (b&e, toward the bottom of the image).

7.3 Summary

In this paper, we proposed denoising and 3D registration algorithms customized for high-resolution mouse brain data from the Knife-Edge Scanning Microscope (KESM). Denoising was based on observed noise characteristics in the KESM data: localized and global FFT-based denoising. Registration in 3D was achieved by the use of a 2D, as-rigid-as-possible deformation that uses Moving Least Squares (MLS). The algorithms were used to register our KESM mouse brain Nissl data set to the rodent standard Waxholm space. Once registered to the standard atlas, we can import a large number of annotations such as the boundary information and text labels of cortical areas and subcortical nuclei, gene expression data, and scientific citation information associated with the specific brain region. Furthermore, high-resolution data from our KESM atlas can also map back to existing atlases, serving as an invaluable resource for neuroscientists.



(a) KESM slice (b) WHS slice (c) Deformation

(d) KESM slice (e) WHS slice (f) Deformation

Figure 7.6: Deformation in the Coronal (Top) and Horizontal (Bottom) Planes with a Uniform Grid. The orange dots in the (a) KESM and (b) WHS slices are the corresponding landmarks. (c) Deformation result of the KESM slice (left) shown with the corresponding WHS slice (right). The grid in the foreground is the estimated deformation grid. Also, the orange dots in the (d) KESM and (e) WHS slices are the corresponding landmarks. (f) Deformation result of the KESM slice (left) shown with the corresponding WHS slice (right).

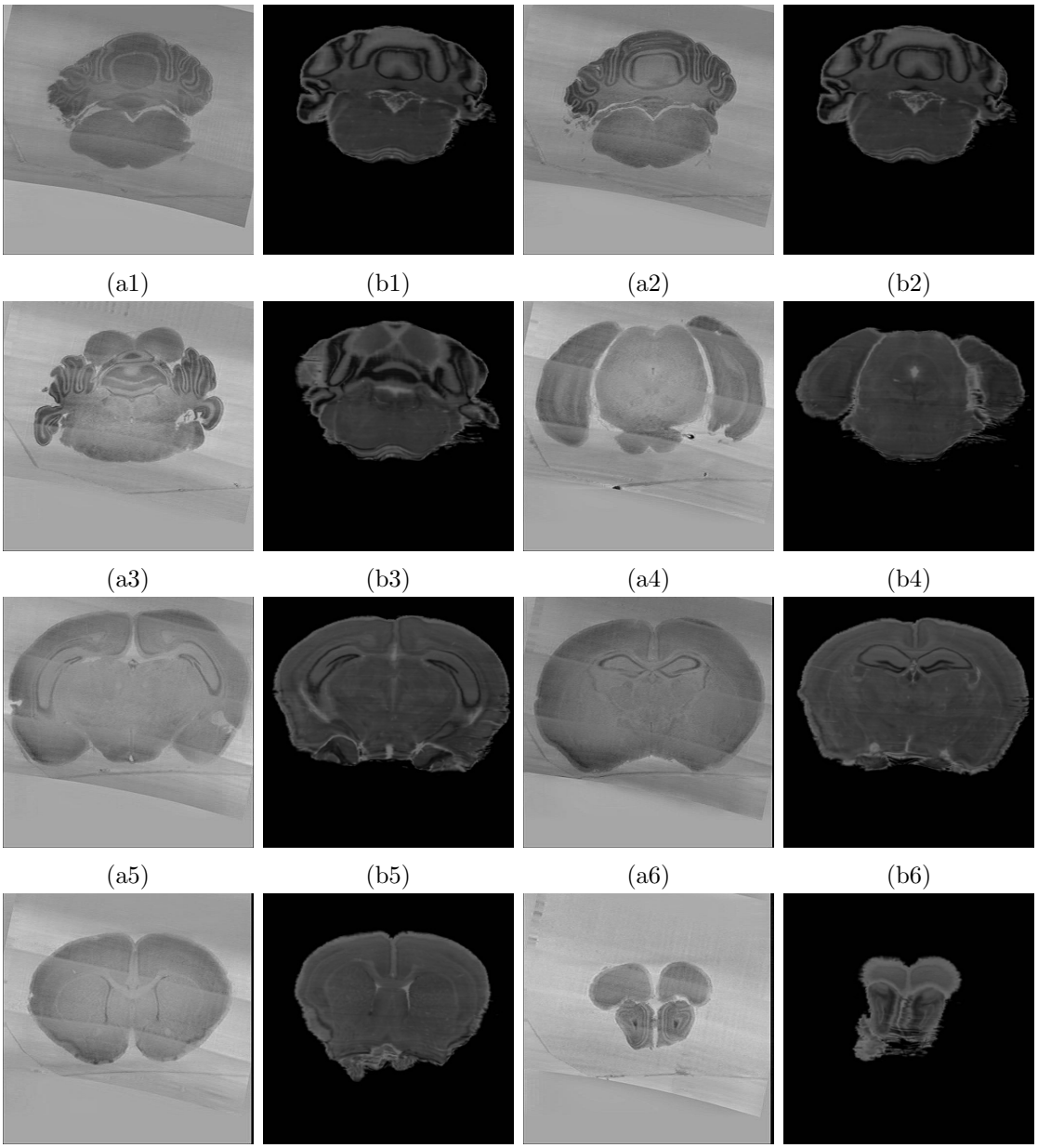
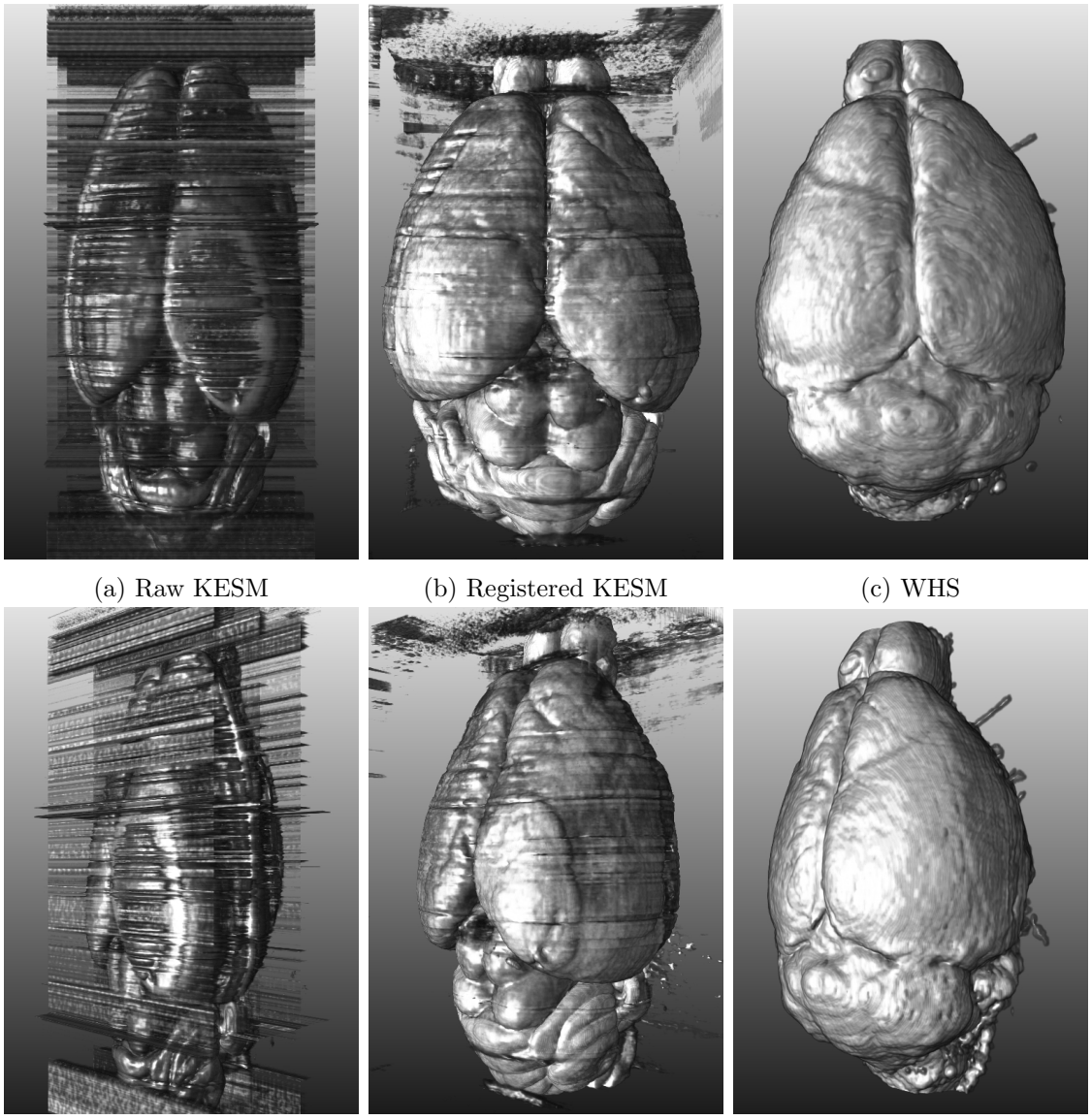


Figure 7.7: Comparison of Denoised KESM Data Registered to the WHS Vs. the Corresponding WHS Data (Coronal Sections). (a1)–(a8): KESM. (b1)–(b8): WHS. See text for details.



(a) Raw KESM (b) Registered KESM (c) WHS

(d) Raw KESM (e) Registered KESM (f) WHS

Figure 7.8: 3D Visualization for Qualitative Comparisons. (a&d) The raw KESM volume stained with Nissl. (b&e) The denoised and registered KESM volume. (c&f) The Nissl-stained optical histology WHS volume. The detailed convolutions in the cerebellar cortex (bottom of the image) can only be seen in our high-resolution KESM atlas.

8. AUTOMATED CELL DETECTION VIA INCREMENTAL LEARNING[†]

Analysis of neuronal distributions in the brain plays an important role in the understanding the organization and in the diagnosis of disorders of the brain. For example, cytoarchitectonics provides deep insights into the cortical map organization, and abnormal growth or reduction in the number of neurons can indicate disorders in the region.

Recent advances in high-throughput 3D microscopy techniques have opened the way to a fully quantitative investigation of neuronal distributions at the whole-brain scale [64, 74, 93, 31]. The Knife-Edge Scanning Microscope (KESM) [16, 55] is one of the first instruments to produce sub-micrometer resolution ($\sim 1\mu\text{m}^3$) data from whole small animal brains. Among our successfully imaged two Golgi (neuronal morphology) [55, 15, 16], India ink (vascular network) [57], and Nissl (soma distribution) [15, 14] data sets, the Nissl data set in particular enables detailed studies of whole-brain cortical and subcortical distribution of neuronal cell bodies.

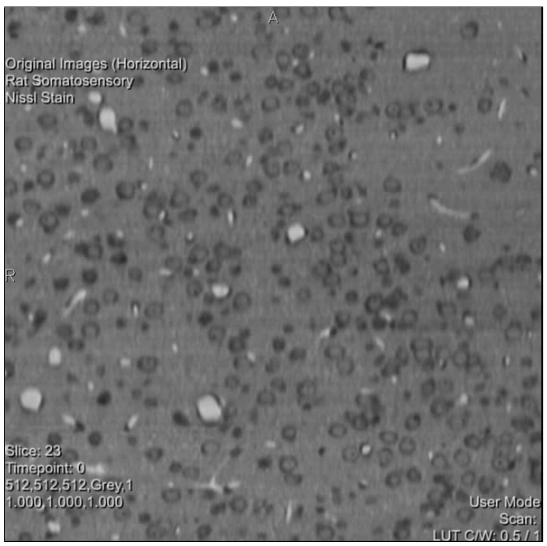
However, a quantitative analysis that seeks to count every neuron in such high-resolution data is faced with a serious challenge. In this project, I introduce a scalable, effective quantitative analysis method for neuron detection using supervised machine learning. An effective learning in such a situation (huge, rapidly growing data) requires: (1) low computational cost (e.g., linear mapping), (2) non-iterative, (3) no accumulation of data points, (4) no retraining, and (5) sufficient accuracy. Mainstream machine learning techniques in the broader category of instance based learning or gradient-based approaches do not meet one or more of these requirements.

[†]Reprinted with permission from "Scalable, Incremental Learning with MapReduce Parallelization for Cell Detection in High-Resolution 3D Microscopy Data" by Sung, Woo, Goodman, Huffman, and Choe, the International Joint Conference on Neural Networks (IJCNN), 2013, the IEEE copyright line © 2013 IEEE.

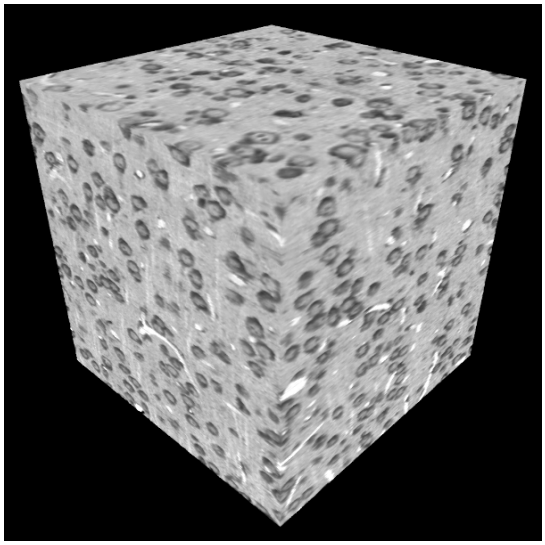
Online learning with stochastic gradient descent addresses most of these issues but it cannot scale up to extremely large data sets.

Here I propose a highly scalable incremental learning algorithm that does not need retraining or retention of old raw data. This algorithm uses Principal Components Analysis (PCA) on its own for classification (i.e., not using other supervised algorithms on top of it; cf. [49], and the Discussion section) to analyze soma distribution in the KESM Nissl-stained three-dimensional (3D) rat brain data set, illustrated in Fig. 8.1. We are able to estimate the true cell density in our data set because every cell should be stained. Synaptic connections cannot be detected because Nissl stain only highlights cell bodies in the brain, whereas the size of individual cells in our high-resolution data set can be used as a possible feature for classification of cell type (e.g., neurons vs. astrocytes).

The main concept of this algorithm is to separate the labeled data set into class-specific subsets and run PCA separately on each subset. This will result in class-specific eigenvector matrices. Given a novel test data, the different eigenvector sets are used to project and in reverse reconstruct the novel data. The class label associated with the eigenvector set that gave the best reconstruction determines the labels of samples in the novel data. This approach is highly scalable, since only the eigenvector matrices need to be stored, and they are orders of magnitude smaller than the raw input data. No retraining is necessary either, since voting or averaging can be used to classify new data samples based on the stored eigenvector matrices. I tested my approach on our KESM rat Nissl data set, showing superior performance compared to an ANN-based benchmark, and scalable learning capabilities. Furthermore, the algorithm is highly parallelizable on both the training and the testing side, thus is easily implementable in parallel data processing frameworks such as MapReduce [17, 72].



A. Nissl image



B. Nissl volume

Figure 8.1: KESM Nissl Data. Nissl-stained tissue data from KESM are shown (rat somatosensory cortex). A. A single image (300 μm wide). B. A 300 μm cube. The dark donut-shaped objects are the cell bodies labeled by Nissl. White ovals are unstained regions representing blood vessels. The voxel resolution was 0.6 μm $\times$ 0.7 μm $\times$ 1.0 μm.

8.1 PCA-based Supervised Learning

My proposed algorithm is illustrated in Fig. 8.2.

Fig. 8.2A shows the training process. The labeled training set $\mathbf{X}$, each row of which is an input vector, is separated into two subsets: cell center class (+) and off-center class (−). PCA is run separately on the class-specific subsets ($\mathbf{X}^+$ and $\mathbf{X}^-$), resulting in two eigenvector matrices, $\mathbf{V}^+$ and $\mathbf{V}^-$.

Fig. 8.2B shows the testing process. For the testing of novel data, each data vector $\mathbf{x}$ is first projected using the two PCA eigenvector matrices, giving projections $\mathbf{y}^+$ and $\mathbf{y}^-$. From these projections, we attempt to reconstruct the original input vector, using the inverse of the eigenvector matrices ($\mathbf{V}^{+T}$ and $\mathbf{V}^{-T}$), producing $\tilde{\mathbf{x}}^+$ and $\tilde{\mathbf{x}}^-$. The class associated with the more accurate reconstruction ($||\mathbf{x}-\tilde{\mathbf{x}}^+||$ vs.

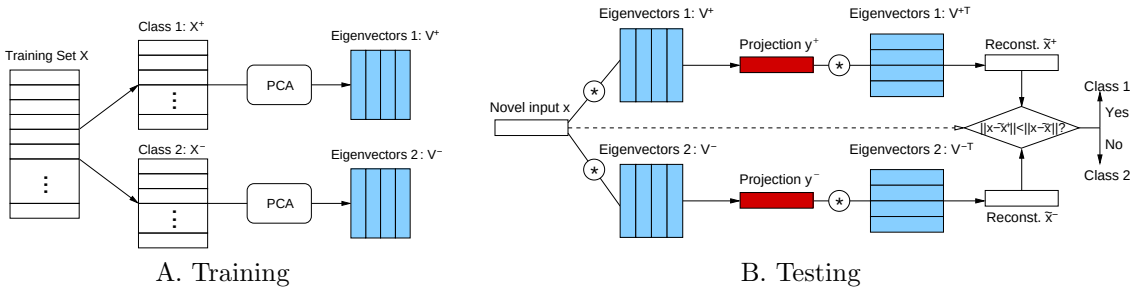


Figure 8.2: PCA-based Supervised Learning. A. Training. The training set $\mathbf{X}$ is separated into two subsets based on the class of each input, and PCA is run separately on these class-specific subsets. B. Testing. For the testing of novel data, the data vector $\mathbf{x}$ is first projected using the two class-specific eigenvector matrices and reconstructed using the inverse of these matrices.

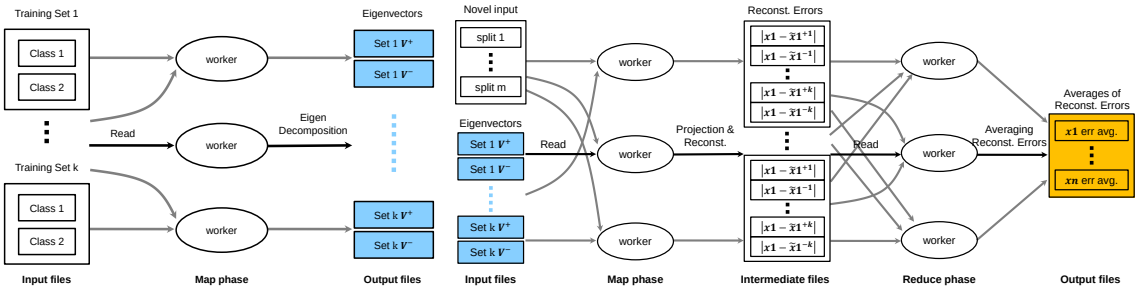
$||\mathbf{x}-\tilde{\mathbf{x}}^{-}||$) determines the label for the new data vector. Not all the eigenvectors need to be used or stored (only a fraction may be necessary), thus making this process computationally cheap. Furthermore, this approach can be implemented with any dimensionality reduction algorithm that allows an inverse mapping, thus it is not limited to PCA.

My approach does not require an iterative learning (e.g., learning rules based on gradient descent) or relearning process since knowledge from earlier batches of data is encoded and stored in the collected eigenvector matrices that are several orders of magnitude smaller than the raw data. Furthermore, our algorithm can be implemented as a sum of convolutions, rendering it ideal for parallelization, and is thus highly scalable. The algorithm can also be implemented for GPGPU and it would scale nicely (i.e., linearly) even when the size of the input vectors is increased.

8.2 MapReduce Parallelization

My algorithm is highly parallelizable due to its incremental nature. To exploit this property, we developed a MapReduce-based implementation of the algorithm.

The MapReduce framework divides parallel data processing tasks into the map phase and the reduce phase, where during the map phase, tasks are divided and results are emitted, and during the reduce phase, the emitted results are sorted and consolidated [17]. MapReduce is a highly effective and popular framework for big data analytics.



A. Training B. Testing

Figure 8.3: MapReduce Model of PCA-based Supervised Learning. A. Training. The k labeled training sets arrive all at once as input, and the *map* function parallelizes PCA computations of the individual training sets. This process does not require a *reduce* function (*reduce* is the identity function). B. Testing. Based on the output files of the training process, the *map* function projects and in turn reconstructs the m splits of the novel data set in parallel. The *reduce* function groups the reconstruction errors by voxel and averages them, producing the voted class results of n data vectors from all voxels.

Fig. 8.3A shows an overview of our MapReduce-based training process. In order to parallelize PCA computations of multiple training sets, we designed a *map* function. The k labeled training sets, each including data vectors from two classes (cell-center [+] and off-center class [-]), are fed as the input files for the *map* function. In the *map* function, we parallelize PCA computations of the class-specific subsets from the training sets, generating two eigenvector matrices per training set: $\mathbf{V}^+$ and $\mathbf{V}^-$. The output files are the eigenvector matrices from the training sets, producing

tuples of the following form:

$$\langle \text{(test set split ID, training set ID, class), eigenvec. matrix} \rangle.$$

Because MapReduce sorts the *map* output values by their keys, we use multiple attributes as a key for grouping in the later stage. For the training process, we do not need a *reduce* function because we already have the eigenvector matrices calculated from the *map* phase (*reduce* is the identity function).

Fig. 8.3B shows an overview of our MapReduce-based testing process. The *map* function of the testing process uses the output files from the training process as the input. To check if a data vector from each voxel in a novel data volume is in the cell-center class, we need to prepare all data vectors from all voxels in the data volume. Instead of collecting data vectors for all voxels of the data volume into a single data set, our *map* function in the testing stage generates one data set per one xy slice in the data volume (i.e., all voxels at a specific depth in z). We call this data set a "split". With each split, we project, reconstruct, and calculate the error using *all* eigenvector matrices collected in the training stage. This *map* phase produces a long list of tuples of the format

$$\langle \text{(voxel coordinate), training set ID, class, reconst. error} \rangle$$

and stores them as intermediate files on the local disks. Then, our *reduce* function takes these intermediate files as an input and groups the results in parallel by their voxel coordinate IDs and averages the class-by-class reconstruction errors of each data vector for each voxel coordinate. Based on the averages of the reconstruction errors, the *reduce* function finalizes the classification of the data vector from each voxel.

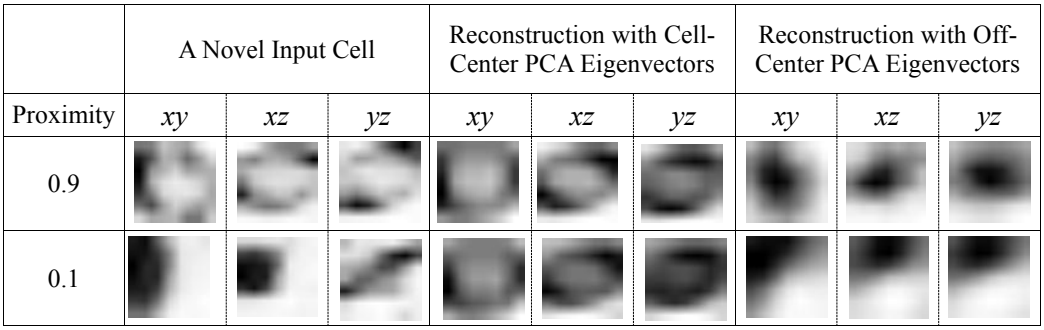


Figure 8.4: Cell Body Reconstruction. Reconstruction of 11 μm cubes: top row = cell center (center proximity value = 0.9/1.0), bottom row = off-center (center proximity value = 0.1/1.0). xy, yz, and xz are the three orthogonal cross-sections through the middle of the small volumes enclosing the labeled position. We can see that the reconstruction based on the matching class is more accurate.

8.3 Experiments and Results

8.3.1 Incremental Learning and Results

I used data acquired with our Knife-Edge Scanning Microscope (KESM). The imaging resolution was 0.6 μm $\times$ 0.7 μm $\times$ 1.0 μm. The particular data set used in this paper was from the rat cerebral cortex stained with Nissl. From the Nissl data set, we made ten subvolumes (200 $\times$ 200 $\times$ 100 voxels each, subsampled down to 100 $\times$ 100 $\times$ 50 voxels for faster computation). Each subvolume contained an average of 200 cell bodies.

For my training set, I labeled the center voxels of the individual neurons and the voxels in off-center regions in selected subvolumes. Then we determined the diameter d (= 11) of the sphere that fits around most cell bodies. Once we figured out the diameter, we extracted three orthogonal cross-sections of size 11 $\times$ 11 (xy, yz, and xz) centered at the labeled voxels to construct the input vectors (Fig. 8.4, left-most column).

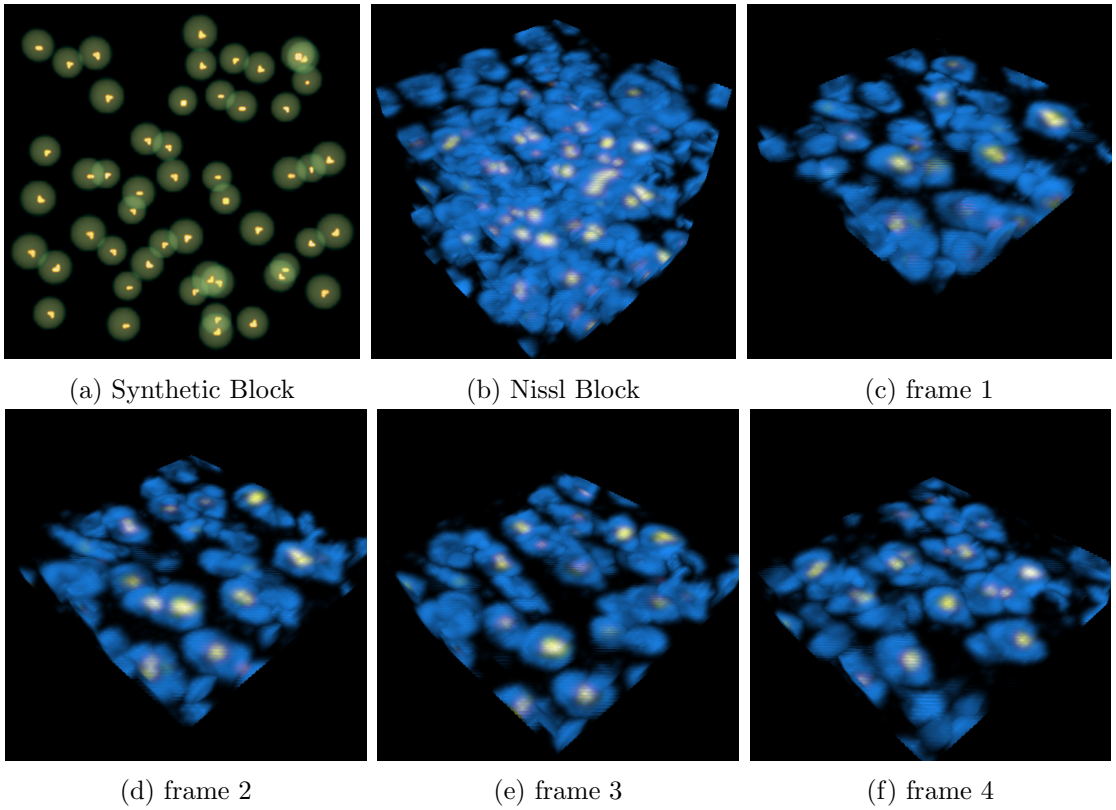
From the training data subvolumes we ran PCA separately on the cell center class and the off-center class to obtain two sets of eigenvectors. We took the first five PCs from the eigenvector set of the cell center class. For the off-center class, we chose three PCs from its eigenvector set. In both cases, the cut-off was determined by the variance accounted for by these PCs.

Given a novel data subvolume, we extracted the cross-sections from all voxel locations to construct new input vectors. These extracted input vectors were projected using both PCA eigenvector matrices ($\mathbf{V}^+$ and $\mathbf{V}^-$) from the training data sets and were reconstructed using the inverse of these matrices. The class label of the eigenvector matrix that gave the best reconstruction was assigned to the novel input vectors. We used Euclidean distance as a reconstruction error metric. Fig. 8.4 shows a reconstruction result of the cross-sections from a cell body.

Fig. 8.5 (a) shows the results from a synthetic data (spheres, green) and the predicted cell centers (yellow), showing a near-perfect match. Fig. 8.5 (b) displays the Nissl data of the test subvolume (blue) and the predicted cell centers (yellow), and Fig. 8.5 (c)-(f) illustrate the details of the Nissl results, sweeping through the volume for an easier visual inspection. Again, the results show a close match.

To quantify the accuracy of the algorithm, we tested all voxels in the test data volume with high proximity values (cell center regions) and with low proximity values (off-center regions), and plotted their reconstruction errors (Euclidean distances). The results are shown in Fig. 8.6 (a). The plot shows the $||\mathbf{x}-\tilde{\mathbf{x}}^+|| - ||\mathbf{x}-\tilde{\mathbf{x}}^-||$, thus a negative value would indicate cell center (close to cell center and far from off-center) and a positive value would indicate off-center (far from cell center and close to off-center).

We also evaluated the classification performance of the algorithm on the test data set. Fig. 8.6 (b) shows the quantitative comparison for the performances of our



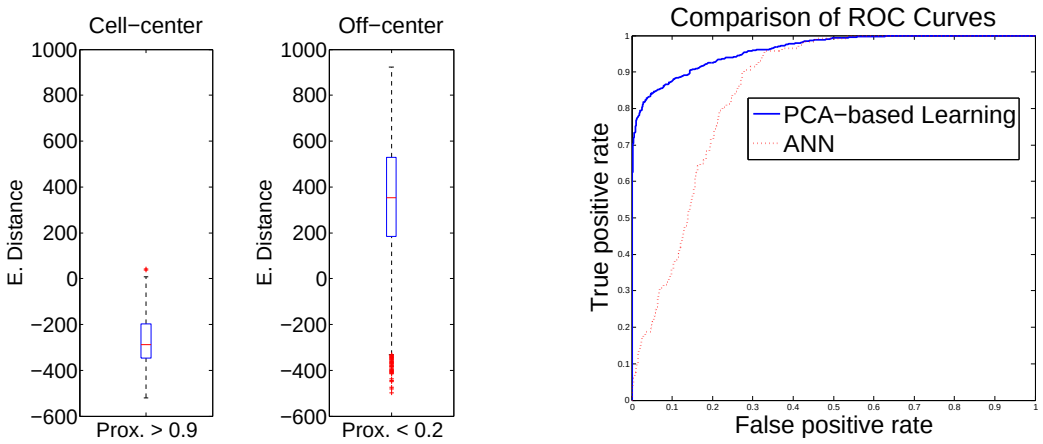
(a) Synthetic Block (b) Nissl Block (c) frame 1

(d) frame 2 (e) frame 3 (f) frame 4

Figure 8.5: Experimental Results. (a) A synthetic volume containing spheres. An overlay of the original data (green) and the predicted cell centers (yellow) shows a near-perfect match. (b) An overview of the cell detection results on the Nissl tissue data (block size = 50 μm cube). Blue displays cell bodies and yellow inside cell bodies the detection results. (c)-(f) Sweeping through the volume with a thin (10 μm) slab shows a close match.

approach against an Artificial Neural Networks (ANN) benchmark [56] using the Receiver Operating Characteristic (ROC) curve. The Area Under Curve (AUC) of the ROC curve data from our approach was 0.9614, compared to 0.8228 from the ANN benchmark. In [56], the ANN approach was found to be superior to Hough-transform and LoG-based blob detection, thus our approach is expected to outperform those as well.

Furthermore, to demonstrate the scalability of our incremental learning algo-



(a) Distance Distribution (b) ROC Curves

Figure 8.6: Accuracy Analysis. (a) The two plots show Euclidean distance distributions of 173 cell center (proximity value $> 0.9/1.0$) and 310,643 off-center (proximity value $< 0.2/1.0$) data points. The distances were computed by $||\mathbf{x}-\tilde{\mathbf{x}}^{+}|| - ||\mathbf{x}-\tilde{\mathbf{x}}^{-}||$, thus, a negative value would indicate cell center and a positive value off-center. The box-whisker plot shows the median, upper and lower quartile, and standard deviation, with outliers. Note the smaller variance in the cell center class due to the stereotypical shape of neurons. (b) The graph shows a comparison of the ROC Curves for the testing data from our approach against that of an ANN-based approach (AUC: our approach = 0.9614, ANN = 0.8228).

rithm, we measured the test results, adding training data sets incrementally. The results are shown in Fig. 8.7. Given three training subvolumes, we first measured the test results with one training subvolume and obtained the AUC of the ROC curve, 0.9584 (green curve, Inc. 1). The training performances of the other two subvolumes were similar to the first set (AUC = 0.9477 and 0.9580). Next, we averaged the Euclidean distances based on the eigenvector matrices obtained from the first and the second training subvolumes. Using the average as a distance measure, we got higher performance (AUC = 0.9652, blue curve, Inc. 2). As expected, when we added results from a third training subvolume (the third subvolume), we found an increased AUC value, 0.9667 (red curve, Inc. 3). Interestingly enough, the AUC of

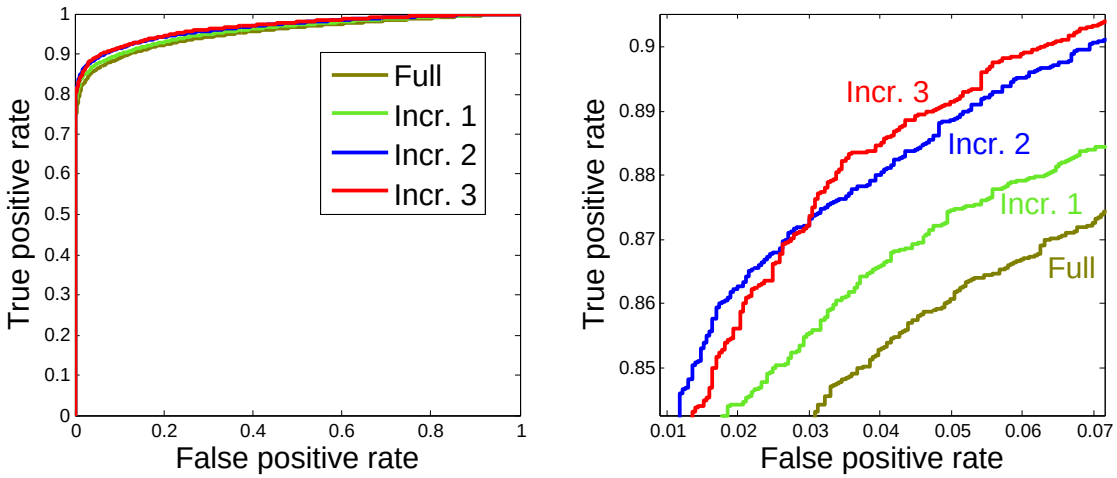


Figure 8.7: Scalability Experiment. ROC curves (left: a full ROC curve, right: zoomed in ROC curve) were computed on a test subvolume to demonstrate the scalability of our incremental learning algorithm. Full: train with 3 subvolumes combined, Incr. 1: train with 1 subvolume, Incr. 2: incrementally learn with 2 subvolumes, Incr. 3: incrementally learn with 3 subvolumes. AUC values were: Full = 0.9526, Incr. 1 = 0.9584, Incr. 2 = 0.9652, Incr. 3 = 0.9667. See text for a detailed discussion.

the test results from training with all three subvolumes together in a single run was 0.9526 (olive curve), and this performance could not go beyond that of Inc. 3 case (red curve), because the performance, at most, reaches the highest AUC value among the training subvolumes. However, Inc. 3 overcame this limitation.

We also experimented with three different training subvolumes that give widely different performances: 0.9681, 0.8497, and 0.8213 in AUC values (some of which may be due to varying levels of noise in the data set caused by the cutting process). Expectedly, when we added training subvolumes incrementally and averaged the test results, the AUC values stayed close to the highest performance (AUC = 0.9422 and 0.9344), even though the AUC values of the two training subvolumes were not high enough. When we trained with the three subvolumes as a single training set, we obtained a better performance (0.9538) compared to the incremental cases. Although

these are mixed results, our approach shows that incremental learning can maintain adequate performance even when certain data batches are of poor quality.

8.3.2 MapReduce Parallelization and Results

Our rat brain Nissl image data set is unstructured and is huge, which is a primary property of big data. Thus, it motivates us to develop our highly scalable algorithm in MapReduce manner which is a major framework for big data analysis. Our experimental results show that our MapReduce approach greatly reduces the computing time while maintaining the same accuracy of the reconstruction.

Our MapReduce algorithm was implemented on Apache Hadoop and we tested our algorithm under Amazon's Elastic Compute Cloud (EC2). Apache Hadoop project has built the Hadoop Distributed File Systems (HDFS) and the Hadoop platform to implement MapReduce. Amazon Web Service (AWS) provides readily available computing nodes where Apache Hadoop platform can be used. Because of such an availability, developers and scientists are able to run their own *map* and *reduce* functions on inexpensive distributed AWS nodes.

For the training process, we used three training volumes (each $100 \times 100 \times 50$ pixels), in each of which the number of data vectors for each class was about 200 (i.e., 200 labeled voxels per class). The input files for the *map* function in this process consisted of the file system paths to the directory containing the training volumes. Once we ran PCA on the class-specific subsets in each volume, we obtained eigenvector matrices of the class-specific subsets. Three sets of eigenvector matrices were generated; a total of six matrices.

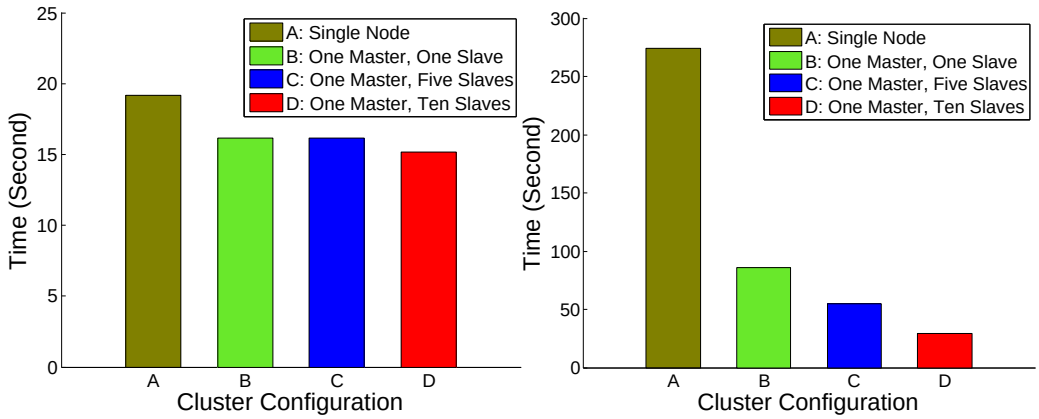
For the testing process, we implemented a *map* function and a *reduce* function. First, we prepared the input files for the testing process based on the result from the training process. The input files included the eigenvector matrices, test set split

ID (depth in z in the data volume), and the class associated with the eigenvector matrix. The input files consisted of 300 tuples (50 splits $\times$ 6 eigenvector matrices). We used these input files for the *map* function. Based on the testing set split IDs, the function could directly access the depth of the testing volume corresponding to each ID and extracted 10,000 data vectors from all voxels at that depth ($= 100 \times 100$). Next, it calculated the projection and in turn reconstruction of the data vectors using the eigenvector matrices in the input files. Finally, it calculated the reconstruction errors, writing them into intermediate files on the disk. The total number of tuples in the intermediate files were about 3,000,000.

In the *reduce* phase, the *reduce* function took the intermediate files as an input, and grouped the reconstruction errors (six per each voxel, corresponding to the six eigenvector matrices from the training phase) by the voxel coordinates. Next, the *reduce* function averaged the class-by-class reconstruction errors and finalized the class assignment of each data vector.

We set up three cluster configurations using Amazon cloud computing EC2 (`http://aws.amazon.com/`): one master node and one slave node; one master node and five slave nodes; and one master node and ten slave nodes. In order to configure the clusters, we equipped the Apache Whirr 0.8.1 libraries. We also used the Apache Hadoop 1.1.1 libraries to run our *map* and *reduce* functions in parallel. For each physical node of EC2, we chose the Ubuntu 12.04 LTS 64-bit operating system, each having quad-core 2xIntel Xeon X5570 CPU and 23.00 GB memory (EC2 instance API name: $cc1.4xlarge$, `http://aws.amazon.com/ec2/instance-types/`). Besides the difference in the number of physical nodes, we assigned a specific number of tasks for each job.

Fig. 8.8A shows the comparison of training process performance under different cluster configurations. We ran the *map* function with 5 tasks in each job. As we



A. Training Performance B. Testing Performance

Figure 8.8: MapReduce Parallelization Performance Comparison. A. The bar plot displays the comparison of training MapReduce process performances under different cluster configurations. The olive bar represents the performance for a single node, the green bar for one-master-and-one-slave nodes, the blue bar for one-master-and-five-slave nodes, and the red bar for one-master-and-ten nodes. Except for the single node case, we ran 5 *map* tasks per job (note that we do not need any *reduce* task). B. The bar plot shows the comparison of testing MapReduce process performances under different cluster configurations. The color keys are the same as A. Except for the single node case, we ran 35 *map* tasks and 10 *reduce* tasks in each.

expected, compared to the performance result under the single node configuration, when we increased the number of physical nodes, the performance of our MapReduce approach improved. However, the performance gain was small, which was expected, because we only had three training sets and the parallelization was only over these three sets.

Fig. 8.8B shows the comparison of testing process performance under different cluster configurations. We ran the *map* function with 35 tasks and *reduce* with 10 tasks in each job. As we expected, compared to the performance result under single node configuration, when we increased the number of physical nodes, the performance improved. Unlike the training process, we noticed that the performance of the testing MapReduce approach was greatly improved (nearly 10 times for the 10 slave node

case) compared to serial computation (Fig. 8.8B). Furthermore, each increment in the number of physical nodes gave consistent improvement in performance.

8.4 Summary

In this paper, we presented a novel scalable incremental learning algorithm for fast quantitative analysis of massive, growing, sparsely labeled data from a high-throughput 3D microscope (the Knife-Edge Scanning Microscope). The approach is based on PCA used on its own for classification: class-specific projection and reconstruction. Our algorithm showed high accuracy, 0.9614 (the AUC of the ROC curve on a test set), compared to an ANN-based benchmark (AUC = 0.8228), demonstrating the scalability of the algorithm by pooling results from a growing data set. Furthermore, we implemented our incremental learning algorithm with MapReduce parallelization to greatly increase the performance in a multiple cluster configuration. We expect our approach to be broadly applicable to the analysis of high-throughput medical imaging data.

9. DISCUSSION

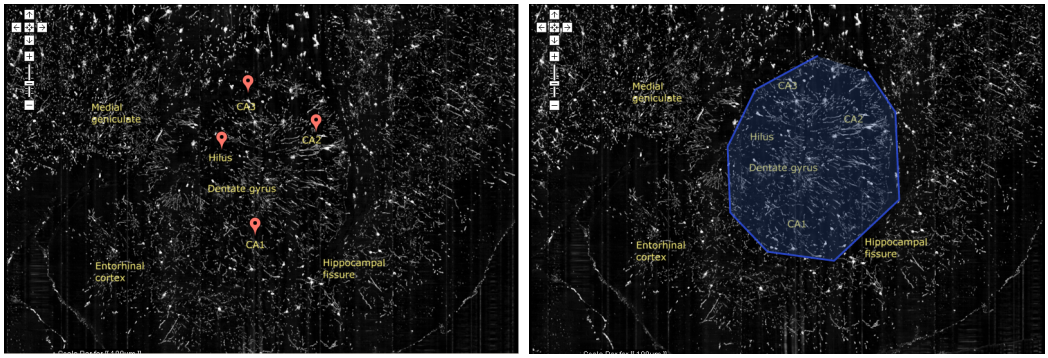
9.1 Exploration

KESMBA provides an effective solution to the visualization and accessibility problems that high-resolution brain atlases confront. The multi-section overlays produce a 3D view to display the structural information of the data, while keeping the computational overhead low by limiting the number of image tiles to download, and enabling a quick identification of the region of interest for full local volume download. This general-purpose property yields broad applicability to other types of data sets from array tomography [63], confocal (and multiphoton) microscopy, or electron microscopy.

To generate a single pre-overlay layer of tiles from multiple sections of tiles is a possible solution to enhance KESMBA loading time. Reducing tile numbers by combining multiple layers will greatly influence the display time. The pre-overlay layer, which has a geometric property itself, supports the identical pseudo 3D view of multiple section overlays, except for coarser z-axis navigation. Fortunately, if such a pre-overlay layer is created at each z-depth, this navigation challenge will be overcome. A simple ImageMagick script can compress multiple tiles into one, using the "composite" command.

To better serve the purpose of information sharing, KESMBA needs to provide richer annotations. For this, allowing a collaborative annotation by experts in various domains will be greatly beneficial. This can be done by combining KESMBA with a *wiki* system, similar to Wikimapia (`http://wikimapia.org/`). Again, we can benefit from the intrinsic features of GoogleMaps or OpenLayers. They support various types of overlay objects which are tied to the map tiles with the latitude and longitude

coordinates. The types of overlay objects include icons, polygons, polylines, and markers. These overlay objects can provide various means to facilitate collaborative user annotations. Fig. 9.1 shows preliminary results of enabling marker and polygon drawing in KESMBA. The annotation can be saved in an XML file whose coordinate values link to the related image tile.



(a) Maker Annotation (b) Polygon Annotation

Figure 9.1: Examples of Graphical Annotation.

Each annotation that a user creates can be visible only to the annotator, to a group or to everyone. With user registration, we can develop a ranking system for users to maintain integrity where each user is assigned a rank and the rank is recorded for every annotation made by the user. User ranking can be managed with a trust-based method where users recognized as experts and those trusted by experts can get assigned higher ranks. We can also provide a filter capable of dynamically selecting more reliable annotations at higher rank levels.

9.2 Registration

The main contribution of our work is (1) the discovery of KESM-specific noise characteristics and a customized denoising algorithm based on localized/global FFT,

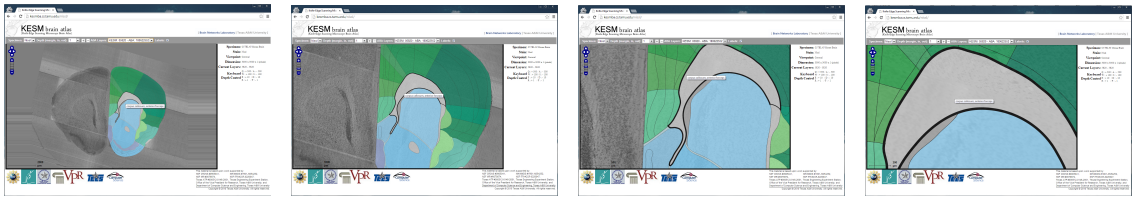
and (2) the use of an as-rigid-as possible deformation method based on MLS for the registration of high-resolution brain atlases to the rodent standard WHS. The technical novelty of our approach, compared to prior works in the field, is the application of FFT-based denoising at two domain-specific spatial scales, and the two-pass (coronal and horizontal) processing of MLS-based deformation for 3D volume data.

Several future directions remain open. First, we need to find slice-by-slice landmarks so that possible local deformations in the tissue (due to the dehydration and polymer embedding process) can be ironed out. Also, additional anatomically correct landmarks need to be selected by an expert neuroanatomist. Furthermore, my landmark-based 3D registration framework cannot provide local region mapping even with a large number of landmarks. Klein et al. [41] surveyed fourteen nonlinear deformation algorithms applied to brain image registration, each of which was applied at least 2,168 times. The authors evaluated these algorithms using 8 different error measures and found that symmetric diffeomorphic registration (SyN) [5] and automatic registration toolbox (ART) packages (`http://www.nitrc.org/projects/art/`) give consistently high-ranking results. In this evaluation study, they also found that the relative performances of the registration methods are little affected by the choice of subject population and labeling protocol. So, even though our brain data modality is different from theirs, I expect SyN and ART to be applicable to KESM data registration.

Furthermore, a neuroinformatics platform needs to be set up to import existing annotations from other mouse brain atlases. With the annotation framework in place, we can utilize existing annotations in the Waxholm Space (WHS) data volumes and in Allen Brain Atlas data volumes. As the KESM data was registered with the WHS as part of my dissertation, we expect this step to be straight-forward. A main research issue is to reciprocate this back to the WHS atlases and to the Allen Brain

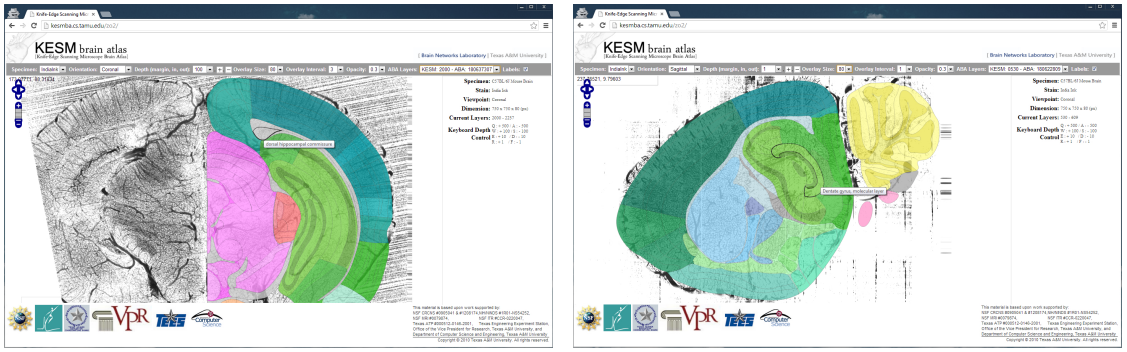
Atlas. We can develop compatible API calls so that those using the WHS atlases and Allen Brain Atlas can import anatomical data from our KESM brain atlas.

The Allen Brain Atlas (ABA) website provides Allen Reference Atlases (ARAs) in the coronal and the sagittal plane to complement a genome-wide map of gene expression in the mouse brain. I developed a crawling tool to extract annotated full-color references of mouse brain structures in SVG format from the ARAs. Once I obtained 132 SVG references and 21 SVG references in the coronal and sagittal plane, I manually mapped the Allen SVG references to our KESM mouse brain atlases and overlayed the Allen SVG references on KESMBA v2. Fig. 9.2 and Fig. 9.3 show the mapped results in KESM Nissl coronal and India ink coronal and sagittal atlases. This overlay plays a crucial role in annotations of our high-resolution moue brain data. We can develop a registration algorithm to map our KESM data to the ABA instead of my manual approach. For automated registration, I expect that the ITK registration framework (`http://www.itk.org/`) can be a possible solution.



A. Zoom level 3 B. Zoom level 4 C. Zoom level 5 D. Zoom level 6

Figure 9.2: Registration of Allen Reference Atlases (ABAs) to KESM Nissl Coronal Atlas. A SVG Allen coronal reference mapping to the multi-scale layers in the KESM Nissl coronal atlas is shown. They are gradually zooming into the Corpus Callosum.



A. ABA over India ink coronal B. ABA over India ink sagittal

Figure 9.3: Registration of Allen Reference Atlases (ABAs) to KESM India Ink Coronal and Sagittal Atlases. A SVG Allen coronal reference mapping to the KESM India ink coronal atlas and a SVG Allen sagittal reference mapping to the KESM India ink sagittal atlas are shown.

9.3 Analysis

Our approach is related to Linear Discriminant Analysis (LDA) [51], but unlike LDA, it does not consider different classes together (between-class scatter matrix). Dimensionality reduction algorithms commonly use reconstruction error as a learning metric but they are not used in a supervised manner because often they do not allow inverse mapping, e.g. Isomap [92]. The approach is also related to committee learning in the sense that the final classification is based on voting, but in our case, we do not have a fixed committee: the committee is continually growing.

Watanabe [99] first introduced a subspace method called Class-Featuring Information Compression (CLAFIC) for pattern classification. To assign new test samples to a class, the CLAFIC method projects the new samples onto class-specific eigenspaces, and the samples are assigned to the class which gives the maximum norm value. Our approach, however, projects and in reverse reconstructs new samples using the class-specific eigenvectors and classify the samples based on the reconstruction errors. Malagón-Borja and Fuentes [49] used the same PCA reconstruction technique for

supervised classification, but not in the context of scalable, incremental learning for massive, growing data.

Several incremental learning methods for eigenspace models have been proposed [10, 12, 29, 28, 70]. All these approaches compute an eigenspace by updating a single eigenspace model as new training samples are made available. In our case, we calculate separate eigenspaces for different batches (subvolumes), but instead of updating a single eigenspace, we keep all the eigenspaces and use them to project and reconstruct new input samples, based on which we take a vote to decide the class.

10. CONCLUSION

In my dissertation, I presented KESMBA, a new web-based mouse brain atlas with improved accessibility and enhanced 3D visualization. KESMBA was designed to facilitate the sharing of the massive high-resolution mouse brain data acquired from the KESM. The multiscale tiles allowed quick and consistent downloading time and the 3D method enabled effective 3D visualization. Moreover, KESMBA allows access from any Internet devices because it minimizes the client-side computation and is implemented using standard Javascript library. KESMBA can serve as an effective and efficient informatics framework for delivering large image volumes to the neuroscience research community.

On top of KESMBA, since the viewpoint of KESMBA is fixed, to better appreciate the full 3D morphology of the objects embedded in the data volumes, I have developed a WebGL-based approach that complements the KESM brain atlas for interactive viewing. It provides not only the geometrical reconstruction feature from a given volume, but also allows users to interactively access the different angles of view of an image volume.

Furthermore, I explained denoising and 3D registration algorithms customized for high-resolution mouse brain data from the Knife-Edge Scanning Microscope (KESM). Denoising was based on observed noise characteristics in the KESM data: localized and global FFT-based denoising. Registration in 3D was achieved by the use of a 2D, as-rigid-as-possible deformation that uses Moving Least Squares (MLS). The algorithms were used to register our KESM mouse brain Nissl data set to the rodent standard Waxholm Space. Once registered to the standard atlas, I can import a large number of annotations such as the boundary information and text labels of cortical

areas and subcortical nuclei, gene expression data, and scientific citation information associated with the specific brain region. Furthermore, high-resolution data from our KESM atlas can also map back to existing atlases, serving as an invaluable resource for neuroscientists.

Lastly, I introduced a novel scalable incremental learning algorithm for fast quantitative analysis of massive, growing, sparsely labeled data from the Knife-Edge Scanning Microscope. The approach is based on PCA used in a supervised manner: class-specific projection and reconstruction. The algorithm showed high accuracy, 0.9614 (the AUC of the ROC curve on a test set), compared to an ANN-based benchmark (AUC = 0.8228), demonstrating the scalability of the algorithm by pooling results from a growing data set. Furthermore, I implemented my incremental learning algorithm with MapReduce parallelization to greatly increase the performance in a multiple cluster configuration.

I expect my frameworks to enable effective exploration and analysis of our KESM data sets. In addition, I expect my approaches to be broadly applicable to the analysis of other high-throughput medical imaging data.

REFERENCES

[1] L. C. Abbott. High-throughput imaging of whole small animal brains with the knife-edge scanning microscope. In *Neuroscience Meeting Planner, Washington, DC: Society for Neuroscience*, 2008. Program No. 504.2.

[2] K. A. Al-Kofahi, S. Lasek, D. H. Szarowski, C. J. Pace, G. Nagy, J. N. Turner, and B. Roysam. Rapid automated three-dimensional tracing of neurons from confocal image stacks. *IEEE Transactions on Information Technology in Biomedicine*, 6:171–187, 2002.

[3] M. Alexa, D. Cohen-Or, and D. Levin. As-rigid-as-possible shape interpolation. In *Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, pages 157–164, 2000.

[4] G. A. Ascoli, D. E. Donohue, and M. Halavi. Neuromorpho.org: A central resource for neuronal morphologies. *The Journal of Neuroscience*, 27(35):9247–9251, 2007.

[5] B. B. Avants, C. L. Epstein, M. Grossman, and J. C. Gee. Symmetric diffeomorphic image registration with cross-correlation: evaluating automated labeling of elderly and neurodegenerative brain. *Medical Image Analysis*, 12(1):26–41, 2008.

[6] S. R. Aylward and E. Bullitt. Initialization, noise, singularities, and scale in height ridge traversal for tubular object centerline extraction. *IEEE Trans. Medical Imaging*, 21(2):61–75, 2002.

[7] R. A. Baldock, J. B. Bard, A. Burger, N. Burton, J. Christiansen, G. Feng, B. Hill, D. Houghton, M. Kaufman, J. Rao, J. Sharpe, A. Ross, P. Stevenson,

S. Venkataraman, A. Waterhouse, Y. Yang, and D. R. Davidson. Emap and emage: a framework for understanding spatially organized data. *Neuroinformatics*, 1(4):309–325, 2003.

[8] P. J. Basser and D. K. Jones. Diffusion-tensor mri: Theory, experimental design and data analysisa technical review. *NMR in Biomedicine*, 15:456–467, 2002.

[9] A. Borsdorf, R. Raupach, T. Flohr, and J. Hornegger. Wavelet based noise reduction in ct-images using correlation analysis. *IEEE Trans. Medical Imaging*, 27(12):1685–1703, 2008.

[10] J. R. Bunch and C. P. Nielsen. Updating the singular value decomposition. *Numerische Mathematik*, 31(2):111–129, 1978.

[11] A. Can, H. Shen, J. N. Turner, H. L. Tanenbaum, and B. Roysam. Rapid automated tracing and feature extraction from retinal fundus images using direct exploratory algorithms. *IEEE Transactions on Information Technology in Biomedicine*, 3:125–138, 1999.

[12] S. Chandrasekaran, B. S. Manjunath, Y. F. Wang, J. Winkeler, and H. Zhang. An eigenspace update algorithm for image analysis. *Graphical Models and Image Processing*, 59(5):321–332, 1997.

[13] Y. Choe, L. C. Abbott, D. Han, P. Huang, J. Keyser, J. Kwon, D. Mayerich, Z. Melek, and B. H. McCormick. *Knife-edge scanning microscopy: High-throughput imaging and analysis of massive volumes of biological microstructures*. Boston, MA: Artech House Publishers, 2008.

[14] Y. Choe, L. C. Abbott, D. E. Miller, D. Han, H.-F. Yang, J. R. Chung, C. Sung, D. Mayerich, J. Kwon, K. Micheva, and S. J. Smith. Multiscale imaging,

analysis, and integration of mouse brain networks. In *Society for Neuroscience*, 2010.

[15] Y. Choe, D. Mayerich, J. Kwon, D. E. Miller, J. R. Chung, C. Sung, J. Keyser, and L. C. Abbott. Knife-edge scanning microscopy for connectomics research. In *Proceedings of the International Joint Conference onNeural Networks*, pages 2258–2265, Piscataway, NJ, 2011. IEEE Press.

[16] J. R. Chung, C. Sung, D. Mayerich, J. Kwon, D. E. Miller, T. Huffman, J. Keyser, L. C. Abbott, and Y. Choe. Multiscale exploration of mouse brain microstructures using the knife-edge scanning microscope brain atlas. *Frontiers in Neuroinformatics*, 5:29, 2011.

[17] Jeffrey Dean and Sanjay Ghemawat. Mapreduce: simplified data processing on large clusters. *Communications of ACM*, 51:107–113, 2008.

[18] W. Denk and H. Horstmann. Serial block-face scanning electron microscopy to reconstruct three-dimensional tissue nanostructure. *PLoS Biology*, 19:e329, 2004.

[19] M. Ellisman and S. Peltier. *Medical data federation: the biomedical informatics research network*. In I. Foster and C. Kesselman (Eds.), The Grid: Blueprint for a New Computing Infrastructure (2nd Ed.). San Francisco, CA: Morgan Kaufmann, 2004.

[20] Daniel Chern-Yeow Eng and Yoonsuck Choe. Stereo pseudo 3D rendering for web-based display of scientific volumetric data. In *Proceedings of the IEEE/EG International Symposium on Volume Graphics*, 2008.

[21] D.C. Van Essen and K. Ugurbil. The future of the human connectome. *Neuroimage*, 62(2):1299–1310, 2012.

[22] F. Faghih and M. Smith. Combining spatial and scale-space techniques for edge detection to provide a spatially adaptive wavelet-based noise filtering algorithm. *IEEE Trans. Image Processing*, 11(9):1062–1071, 2002.

[23] A. F. Frangi, W. J. Niessen, R. M. Hoogeveen, T. V. Walsum, and M. A. Viergever. Model-based quantification of 3-D magnetic resonance angiographic images. *IEEE Transactions on Medical Imaging*, 18:946–956, 1999.

[24] K. Franklin and G. Paxinos. *The mouse brain in stereotaxic coordinates*. San Diego, CA: Academic Press, 1997.

[25] Dennis F. Galletta, Raymond Henry, Scott McCoy, and Peter Polak. Web site delays: How tolerant are users? *Journal of the Association for Information Systems*, 5(1):1–28, 2004.

[26] B. Hagberg. Defects of immediate memory related to the cerebral blood flow distribution. *Brain and Language*, 5(3):366–377, 1978.

[27] P. Hagmann, M. Kurant, X. Gigandet, P. Thiran, V. J. Wedeen, R. Meuli, and J. P. Thiran. Mapping human whole-brain structural networks with diffusion mri. *PLoS ONE*, 2(7):e597, 2007.

[28] P. Hall, D. Marshall, and R. Martin. Adding and subtracting eigenspaces with eigenvalue decomposition and singular value decomposition. *Image and Vision Computing*, 20(13-14):1009–1016, 2002.

[29] P. M. Hall, D. Marshall, and R. R. Martin. Incremental eigenanalysis for classification. In *British Machine Vision Conference*, 1998.

[30] M. Hawrylycz, R. A. Baldock, A. Burger, T. Hashikawa, G. A. Johnson, M. Martone, L. Ng, C. Lau, S. D. Larson, J. Nissanov, L. Puelles, S. Ruffins,

F. Verbeek, I. Zaslavsky, and J. Boline. Digital atlasing and standardization in the mouse brain. *PLoS Comput. Biol.*, 7(2):e1001065, 2011.

[31] Kenneth Hayworth. Automated creation and SEM imaging of Ultrathin Section Libraries: Tools for large volume neural circuit reconstruction. In *Society for Neuroscience Abstracts*. Washington, DC: Society for Neuroscience, 2008. Program No. 504.4.

[32] H. Hintiryan, L. Gou, B. Zingg, S. Yamashita, H. M. Lyden, M. Y. Song, A. K. Grewal, X. Zhang, A. W. Toga, and H. Dong. Comprehensive connectivity of the mouse main olfactory bulb: analysis and online digital atlas. *Frontiers in Neuroanatomy*, 6:30, 2012.

[33] E. Hodneland, N. V. Bukoreshtliev, T. W. Eichler, X. C. Tai, S. Gurke, A. Lundervold, and H. H. Gerdes. A unified framework for automated 3-d segmentation of surface-stained living cells and a comprehensive segmentation evaluation. *IEEE Trans. Medical Imaging*, 28(5):720–738, 2009.

[34] T. Igarashi, T. Moscovich, and J. F. Hughes. As-rigid-as-possible shape manipulation. *ACM Transactions on Graphics*, 24(3):1134–1141, 2005.

[35] V. Jain, H. S. Seung, and S. C. Turaga. Machines that learn to segment images: a crucial technology for connectomics. *Current Opinion in Neurobiology*, 20(5):653–666, 2010.

[36] F. C. Joelving, R. Billeskov, J. R. Christensen, M. West, and B. Pakkenberg. Hippocampal neuron and glial cell numbers in parkinson's disease-a stereological study. *Hippocampus*, 16(10):826–833, 2006.

[37] G. A. Johnson, A. Badea, J. Brandenburg, G. Cofer, B. Fubara, S. Liu, and J. Nissanov. Waxholm space: an image-based reference for coordinating mouse

brain research. *NeuroImage*, 53(2):365–372, 2010.

[38] A. R. Jones, C. C. Overly, and S. M. Sunkin. The allen brain atlas: 5 years and beyond. *Nature Reviews Neuroscience*, 10:821–828, 2009.

[39] E. Jurrus, A. R.C. Paiva, S. Watanabe, J. R. Anderson, B. W. Jones, R. T. Whitaker, E. M. Jorgensen, R. E. Marc, and T. Tasdizen. Detection of neuron membranes in electron microscopy images using a serial neuralnetwork architecture. *Medical Image Analysis*, 14(6):770–783, 2010.

[40] M. Kersten, J. Stewart, N. Troje, and R. Ellis. Enhancing depth perception in translucent volumes. *IEEE Transactions on Visualization and Computer Graphics*, 12(5):1117–1124, 2006.

[41] A. Klein, J. Andersson, B. A. Ardekani, J. Ashburner, B. Avants, M. C. Chiang, G. E. Christensen, D. L. Collins, J. Gee, P. Hellier J. H. Song, M. Jenkinson, C. Lepage, D. Rueckert, P. Thompson, T. Vercauteren, R. P. Woods, J. J. Mann, and R. V. Parsey. Evaluation of 14 nonlinear deformation algorithms applied to human brain mri registration. *Neuroimage*, 46(3):786–802, 2009.

[42] Randal A. Koene, Betty Tijms, Peter van Hees, Frank Postma, Alexander de Ridder, Ger J. A. Ramakers, Jaap van Pelt, and Arjen van Ooyen. NETMORPH: A framework for the stochastic generation of large scale neuronal networks with realistic neuron morphologies. *Neuroinformatics*, 7:1539–2791, 2009.

[43] J. Kwon, Y. Choe, and B. H. Mccormick. Automated lateral sectioning for knife-edge scanning microscopy. In *IEEE International Symposium on Biomedical Imaging*, 2008.

[44] C. Lau, L. Ng, C. Thompson, S. Pathak, L. Kuan, A. Jones, and M. Hawrylycz. Exploration and visualization of gene expression with neuroanatomy in the adult mouse brain. *BMC Bioinformatics*, 9:153, 2008.

[45] David Lesage, Elsa D. Angelini, Isabelle Bloch, and Gareth Funka-Lea. A review of 3d vessel lumen segmentation techniques: Models, features and extraction schemes. *Medical Image Analysis*, 13:819–845, 2009.

[46] Anan Li, Hui Gong, Bin Zhang, Qingdi Wang, Cheng Yan, Jingpeng Wu, Qian Liu, Shaoqun Zeng, and Qingming Luo. Micro-optical sectioning tomography to obtain a high-resolution atlas of the mouse brain. *Science*, 330:1404–1408, 2010. See the commentary by Mayerich et al. online: `http://www.sciencemag.org/content/330/6009/1404/reply`.

[47] J. Luisi, A. Narayanaswamy, Z. Galbreath, and B. Roysam. The farsight trace editor: An open source tool for 3-d inspection and efficient pattern analysis aided editing of automated neuronal reconstructions. *Neuroinformatics*, 9(2-3):305–315, 2011.

[48] A. J. MacKenzie-Graham, E.-F. Lee, I. D. Dinov, H. Yuan, R. E. Jacobs, and A. W. Toga. Multimodal, multidimensional models of mouse brain. *Epilepsia*, 48(4):75–81, 2007.

[49] Luis Malagón-Borja and Olac Fuentes. Object detection using image reconstruction with PCA. *Image and Vision Computing*, 27:2–9, 2009.

[50] R. Manniesing, M.A. Viergever, and W.J. Niessen. Vessel axis tracking using topology constrained surface evolution. *IEEE Trans. Medical Imaging*, 26(3):309–316, 2007.

[51] A. M. Martinez and A. C. Kak. PCA versus LDA. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23:228–233, 2001.

[52] M. E. Martone, A. Gupta, M. Wong, X. Qian, G. Sosinsky, B. Ludscher, and M. H. Ellisman. A cell-centered database for electron tomographic data. *Journal of Structural Biology*, 138(1-2):145–155, 2002.

[53] M. E. Martone, J. Tran, W. W. Wong, J. Sargis, L. Fong, S. Larson, S. P. Lamont, A. Gupta, and M. H. Ellisman. The cell centered database project: An update on building community resources for managing and sharing 3d imaging data. *Journal of Structural Biology*, 161(3):220–231, 2008.

[54] E. M. Maryann, A. Gupta, and M. H. Ellisman. E-neuroscience: challenges and triumphs in integrating distributed data from molecules to brains. *Nature Neuroscience*, 7:467–472, 2004.

[55] D. Mayerich, L. C. Abbott, and B. H. McCormick. Knife-edge scanning microscopy for imaging and reconstruction of three-dimensional anatomical structures of the mouse brain. *Journal of Microscopy*, 231:134–143, 2008.

[56] D. Mayerich, J. Kwon, A. Panchal, J. Keyser, and Y. Choe. Fast cell detection in high-throughput imagery using GPU-accelerated machine learning. In *IEEE International Symposium on Biomedical Imaging*, 2011.

[57] D. Mayerich, J. Kwon, C. Sung, L. C. Abbott, J. Keyser, and Y. Choe. Fast macro-scale transmission imaging of microvascular networks using kesm. *Biomedical Optics Express*, 2(10):2888–2896, 2011.

[58] D. Mayerich, B. H. McCormick, and J. Keyser. Noise and artifact removal in knife-edge scanning microscopy. In *Proceedings of IEEE International Symposium on Biomedical Imaging*, pages 556–559, 2007.

[59] B. H. McCormick. System and method for imaging an object, 2004. USPTO patent #US 6,744,572 (for Knife-Edge Scanning; 13 claims).

[60] B. H. McCormick, L. C. Abbott, D. M. Mayerich, , J. Keyser, Jaerock Kwon, Zeki Melek, and Y. Choe. Full-scale submicron neuroanatomy of the mouse brain. In *Society for Neuroscience Abstracts*. Washington, DC: Society for Neuroscience, 2006. Program No. 694.5. http://www.abstractsonline.com/viewer/viewAbstract.asp?akey=3A7DC0B9-D787-44AA-BD08-FA7BB2FE9004&ckey=FE87D85C-A6ED-44BD-8350-DB6D21E3467E&mkey=D1974E76-28AF-4C1C-8AE8-4F73B56247A7&skey=CF70AC54-153B-4559-80B5-69310FB25373.

[61] B. H. McCormick and D. M. Mayerich. Three-dimensional imaging using Knife-Edge Scanning Microscope. *Microscopy and Microanalysis*, 10 (Suppl. 2):1466–1467, 2004.

[62] Bruce H. McCormick. The knife-edge scanning microscope. Technical report, Department of Computer Science, Texas A&M University, 2003. http://research.cs.tamu.edu/bnl/.

[63] K. D. Micheva and S. J. Smith. Array tomography: A new tool for imaging the molecular architecture and ultrastructure of neural circuits. *Neuron*, 55:25–36, 2007.

[64] Kristina Micheva and Stephen J. Smith. Array tomography: A new tool for imaging the molecular architecture and ultrastructure of neural circuits. *Neuron*, 55:25–36, 2007.

[65] S. Mikula, I. Trotts, J. M. Stone, and E. G. Jones. Internet-enabled high-resolution brain mapping and virtual microscopy. *Neuroimage*, 35:9–15, 2007.

[66] P. P. Mitra, M. G. Rosa, and H. J. Karten. Panoptic neuroanatomy: Digital microscopy of whole brains and brain-wide circuit mapping. *Brain, Behavior and Evolution*, 81(4):203–205, 2013.

[67] B. Münch, P. Trtik, and F. Marone. Stripe and ring artifact removal with combined wavelet - fourier filtering. *Optics Express*, 17(10):8567–8591, 2009.

[68] L. Ng, A. Bernard, C. Lau, C. C. Overly, H.-W. Dong, C. Kuan, S. Pathak, S. M. Sunkin, C. Dang, J. W. Bohland, H. Bokil, P. P. Mitra, L. Puelles, J. Hohmann, D. J. Anderson, E. S. Lein, A. R. Jones, and M. Hawrylycz. An anatomic gene expression atlas of the adult mouse brain. *Nature Neuroscience*, 12(3):356–362, 2009.

[69] N. Otsu. A threshold selection method from gray-level histograms. *IEEE Transactions on Systems, Man, and Cybernetics*, 9(1):62–66, 1979.

[70] S. Ozawa, S. Pang, and N. Kasabov. Incremental learning of chunk data for online pattern classification systems. *IEEE Transactions on Neural Networks*, 19(6):1061–1074, 2008.

[71] K. Palágyi, J. Tschirren, E. A. Hoffman, and M. Sonka. Quantitative analysis of pulmonary airway tree structures. *Computers in Biology and Medicine*, 36(9):974–996, 2006.

[72] Biswanath Panda, Joshua S. Herbach, Sugato Basu, and Roberto J. Bayardo. Planet: Massively parallel learning of tree ensembles with mapreduce. In *Proceedings of the 35th International Conference on Very Large Data Bases (VLDB-2009)*, 2009.

[73] J. P. W. Pluim, J. B. A. Maintz, and M. A. Viergever. Mutual-information-based registration of medical images: a survey. *IEEE Transactions on Medical*

Imaging, 22(8):986–1004, 2003.

[74] Timothy Ragan, Lolahon R. Kadiri, Kannan Umadevi Venkataraju, Karsten Bahlmann, Jason Sutin, Julian Taranda, Ignacio Arganda-Carreras, Yongsoo Kim, H. Sebastian Seung, and Pavel Osten. Serial two-photon tomography for automated *ex vivo* mouse brain imaging. *Nature Methods*, 9:255–258, 2012.

[75] M. D. Robinson, C. A. Toth, J. Y. Lo, and S. Farsiu. Efficient fourier-wavelet super-resolution. *IEEE Trans. Image Processing*, 19(10):2669–2681, 2010.

[76] A. Roebroeck, R. Galuske, E. Formisano, O. Chiry, H. Bratzke I. Ronen, D-S Kim, and R. Goebel. High-resolution diffusion tensor imaging and tractography of the human optic chiasm at 9.4 t. *Neuroimage*, 39(1):157–168, 2008.

[77] S. Saalfeld, A. Cardona, V. Hartenstein, and P. Tomančák. Catmaid: collaborative annotation toolkit for massive amounts of image data. *Bioinformatics*, 25:1984–1986, 2009.

[78] S. Saalfeld, A. Cardona, V. Hartenstein, and P. Tomančák. As-rigid-as-possible mosaicking and serial section registration of large sstem datasets. *Bioinformatics*, 26(12):i57–i63, 2010.

[79] A. Sandberg and N. Bostrom. Whole brain emulation: a roadmap. Technical Report 2008-3, Future of Humanity Institute, Oxford University, 2008.

[80] S. Schaefer, T. McPhail, and J. Warren. Image deformation using moving least squares. *ACM Transactions on Graphics*, 25(3):533–540, 2006.

[81] P. Selinger. Potrace: a polygon-based tracing algorithm, 2003. http://potrace.sourceforge.net/.

[82] Paula R. Selvidge, Barbara Chaparro, and Gregory T. Bender. The world wide wait: Effects of delays on user performance. *International Journal of Industrial Ergonomics*, 29(1):15–20, 2002.

[83] G. M. Shepherd, editor. *The Synaptic Organization of the Brain.* New York: Oxford University Press, 5th edition, 2003.

[84] M. Sofka and C.V. Stewart. Retinal vessel centerline extraction using multiscale matched filters, confidence and edge measures. *IEEE Trans. Medical Imaging*, 25(12):1531–1546, 2006.

[85] O. Sporns. The human connectome: a complex network. *Annals of the New York Academy of Sciences*, 1224:109–125, 2011.

[86] O. Sporns. *Networks of the Brain.* Cambridge, MA: MIT Press, 2011.

[87] O. Sporns, G. Tononi, and R. Kötter. The human connectome: A structural description of the human brain. *PLoS Computational Biology*, 1(4):e42, 2005.

[88] J. Starck, E. J. Candes, and D. L. Donoho. The curvelet transform for image denoising. *IEEE Trans. Image Processing*, 11(6):670–684, 2002.

[89] Ying Sun. Automated identification of vessel contours in coronary arteriograms by an adaptive tracking algorithm. *IEEE Trans. Medical Imaging*, 8:78–88, 1989.

[90] S. M. Sunkin, L. Ng, C. Lau, T. Dolbeare, T. L. Gilbert, C. L. Thompson, M. Hawrylycz, and C. Dang. Allen brain atlas: an integrated spatio-temporal portal for exploring the central nervous system. *Nucleic Acids Research*, 41 (D1):D996–D1008, 2013.

[91] J. Talairach and P. Tournoux. *Co-planar stereotaxic atlas of the human brain.* New York: Thieme, 1988.

[92] J. B. Tenenbaum, V. de Silva, and J. C. Langford. A global geometric framework for nonlinear dimensionality reduction. *Science*, 290:2319–2323, 2000.

[93] Philbert S. Tsai, Beth Friedman, Agustin I. Ifarraguerri, Beverly D. Thompson, Varda Lev-Ram, Chris B. Schaffer, Qing Xiong, Roger Y. Tsien, Jeffrey A. Squier, and David Kleinfeld. All-optical histology using ultrashort laser pulses. *Neuron*, 39:27–41, 2003.

[94] D. S. Tuch, T. G. Reese, M. R. Wiegell, and V. J. Wedeen. Diffusion mri of complex neural architecture. *Neuro*, 40:885–895, 2003.

[95] P. J. Uhlhaas and W. Singer. Neural synchrony in brain disorders: Relevance for cognitive dysfunctions and pathophysiology. *Neuron*, 52(1):155–168, 2006.

[96] J. van Pelt and H. Uylings. Natural variability in the geometry of dendritic branching patterns. In K. Lindsay, R. Poznanski, G. Reeke, J. Rosenberg, and O. Sporns, editors, *Modeling in the Neurosciences*, chapter 4. London: CRC Press, 2nd edition, 2005.

[97] S. Venkataraman, P. Stevenson, Y. Yang, L. Richardson, N. Burton, T. P. Perry, P. Smith, R. A. Baldock, D. R. Davidson, and J. H. Christiansen. Emage-edinburgh mouse atlas of gene expression: 2008 update. *Nucleic Acids Research*, 36:D860–D865, 2008.

[98] T. Walter, D. W. Shattuck, R. Baldock, M. E. Bastin, A. E. Carpenter, S. Duce, J. Ellenberg, A. Fraser, N. Hamilton, S. Pieper, M. A. Ragan, J. E. Schneider, P. Tomancak, and J. Hrich. Visualization of image data from cells to organisms. *Nature Methods*, 7:S26–S41, 2010.

[99] Satosi Watanabe. *Knowing and Guessing: Quantitative Study of Inference and Information*. New York: Wiley, 1969.

[100] O. Wink, W. J. Niessen, and M. A. Viergever. Fast delineation and visualization of vessels in 3-d angiographic images. *IEEE Trans. Medical Imaging*, 19:337–346, 2000.

[101] Y. Xu, J. B. Weaver, D. M. Healy, and J. Lu. Wavelet transform domain filters: a spatially selective noise filtration technique. *IEEE Trans. Image Processing*, 3(6):747–758, 1994.