KNIFE-EDGE SCANNING MICROSCOPE MOUSE BRAIN ATLAS IN VECTOR GRAPHICS FOR ENHANCED PERFORMANCE

A Thesis

by

JINHO CHOI

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

MASTER OF SCIENCE

Approved by:

Chair of Committee,	Yoonsuck Choe
Committee Members,	John Keyser
	Louise Abbott
Department Head,	Duncan M. "Hank" Walker

August 2013

Major Subject: Computer Science

ABSTRACT

The microstructure of the brain at the cellular level provides crucial information for the understanding of the function of the brain. A large volume of high-resolution brain image data from 3D microscopy is an essential resource to study detailed microstructures of the brain. Accordingly, we have worked on obtaining high-resolution image data of entire mouse brains using the Knife-Edge Scanning Microscope (KESM). Furthermore, to disseminate these high-resolution whole mouse brain data sets to the neuroscience research community, we developed a web-based brain atlas, the KESM Brain Atlas (KESMBA). To visualize the data sets in 3D while using only a standard web browser, we employed distance attenuation and Google Maps API. The KESMBA is a powerful tool to analyze and share the KESM mouse brain data sets, but the image loading was slow because of the number of raster image (PNG) tiles and the file size. Moreover, since Google Maps API is governed by a commercial license, it does not provide enough flexibility for customization, extension, and mirroring.

To solve these issues, we designed and developed a new KESM mouse brain atlas that uses a vector graphics format called Scalable Vector Graphics (SVG) instead of PNG, and OpenLayers API instead of Google Maps API. The SVG-based KESMBA using OpenLayers allows faster navigation and exploration of the KESM data, and more overlay of layers with the 4 times reduced file size compared to PNG tiles. Due to the reduced file size, the SVG-based KESMBA using OpenLayers is 2.45 times faster than the original atlas. By enhancing the performance, the users can more easily access the KESM data. We expect the SVG-based KESMBA to accelerate new discoveries in neuroscience.

To my wife and son

ACKNOWLEDGEMENTS

I would like to thank my advisor, Dr. Yoonsuck Choe, for his steadfast support and guidance during my masters degree program. Also, I thank my committee members, Dr. John Keyser and Dr. Louise Abbott, for their comments and advice. I am really thankful to Jaewook Yoo for his great help whenever I was in need. And, thanks to my friends and colleagues, including Chul Sung, Suinn Park, and Youngkwon Cha.

I also would like to extend my gratitude to my sponsor, Republic of Korea Army, for affording a chance to have a great experience at Texas A&M University. Furthermore, this research was funded by the National Science Foundation Collaborative Research in Computational Neuroscience project (#1208174, #0905041).

Finally, I am absolutely grateful to my wife, Hawjung Woo, my son, Soya, my parents, and parents-in-law for their patience and love.

TABLE OF CONTENTS

LIST OF FIGURES

FIGURE Page

LIST OF TABLES

TABLE Page

1. INTRODUCTION

The microstructure of the brain at the cellular level provides crucial information for the understanding of the function of the brain. Neuroscientists have been trying to investigate microstructures of the brain to study the construction of brain networks and the interconnections of each neuron [12]. To discern the microstructure of the brain at the cellular level, high-resolution brain images are needed. Meanwhile, whole brain data sets are essential for the study of the connectome, which is the complete wiring diagram of the brain [7], [13]. Recently, advances in technology overcame a challenge of the past that previously restricted brain image data acquisition to small volume with high-resolution or large volume with low-resolution but not both [24].

A large volume of high-resolution brain image data from 3D microscopy is an essential resource to study the microstructures of the brain. Accordingly, we have worked on obtaining high-resolution image data of entire mouse brains using the Knife-Edge Scanning Microscope (KESM) [23]. The data sets include those of brains stained with Golgi (neuronal circuits), Nissl (soma distribution), and India ink (vascular network), imaged at $0.6\mu m \times 0.7\mu m \times 1.0\mu m$ voxel resolution [6]. These KESM data sets are huge in their size (several tera voxels) and are at submicrometer resolution, which is enough to reconstruct cellular and vascular microstructures in detail in all three dimensions. Thus, the KESM data sets allow 3D reconstruction for extracting full-scale cellular and vascular microstructures from the whole mouse brain, which will allow us to answer various open questions in neuroscience [21].

To disseminate these high-resolution whole mouse brain data sets to the neuroscience research community, we developed a web-based brain atlas using the Google Maps API, the KESM Brain Atlas (KESMBA) [2]. To visualize the data sets in

3D while using only a standard web browser on machines without high-performance graphics devices and add-on applications, we employed distance attenuation by overlaying Portable Network Graphics (PNG) images with the background made transparent [10]. However, image loading is slow because of the use of the PNG format, which is larger in file size than other raster images due to loss-less compression, and also due to the added alpha channel in each image (for transparency). Moreover, since Google Maps API is governed by a commercial license [11], it does not provide enough flexibility for customization, extension, and mirroring.

To solve these issues, we designed and developed a new KESM mouse brain atlas that uses a vector graphics format called Scalable Vector Graphics (SVG) instead of PNG, and OpenLayers API instead of Google Maps API. The SVG format is a type of Extensible Markup Language (XML), and each object in the image is recorded as a SVG path element [32]. Due to this, the file size of the mouse brain image data sets can be dramatically reduced when existing raster data (PNG) are converted to the vector graphics format. With the reduced file size, the SVG-based brain atlas decreased the image loading time. Furthermore, we employed an open-source mapping JavaScript library, OpenLayers API [29], to visualize the KESM data sets on the web in place of the Google Maps API. Just like Google Maps API, OpenLayers API provides essential functions for visualizing and navigating geographic data such as overlaying, zooming, and panning. Since OpenLayers API is under a FreeBSD (Berkeley Software Distribution) license, which can be extended and modified by anyone, we can fully customize the API based on our requirements [33].

To evaluate the effectiveness of the SVG-based brain atlas, we used two performance measures. First, we measured the size of the SVG mouse brain data set and compare it with that of the existing PNG data set. Next, we collected the data download and navigation speeds in the new SVG-based brain atlas and compared

them to the previous PNG-based version.

In this thesis, I explain the Knife-Edge Scanning Microscope (KESM), which is the foundation of our research in the Background (Chapter 2) and Related Work chapters (Chapter 3). In Chapter 4, Prior Work: KESM Brain Atlas (KESMBA), I describe the basic concept of KESMBA and methods it used such as Google Maps API, distance attenuation, and multiscale zoom levels. Next, in Chapter 5, I present the KESMBA in Vector Graphics using OpenLayers API. Then, in Chapter 6, I provide detailed implementation methods and the performance results using two metrics: file size and image loading speed. Finally, I discuss contributions, limitations, and future work in Chapter 7; and conclude in Chapter 8.

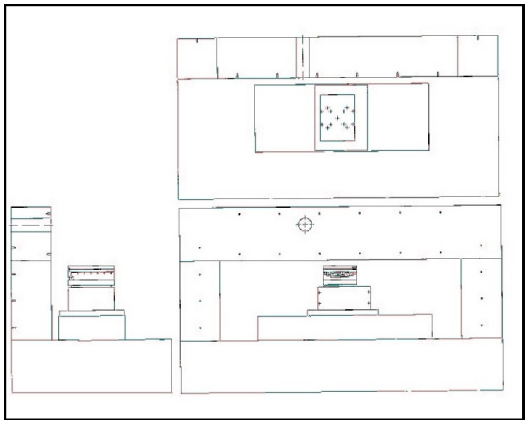
2. BACKGROUND: KNIFE-EDGE SCANNING MICROSCOPE (KESM)

A large volume of high-resolution brain image data from a 3D microscopy is an essential resource to study more realistic microstructures of the brain. Accordingly, at the Brain Network Laboratory (BNL) of Texas A&M, we have worked on obtaining high-resolution image data of entire mouse brains using the Knife-Edge Scanning Microscope (KESM) that is a unique high-throughput and 3D physical-sectioning imaging machine with high-resolution microscopy [21], [8].

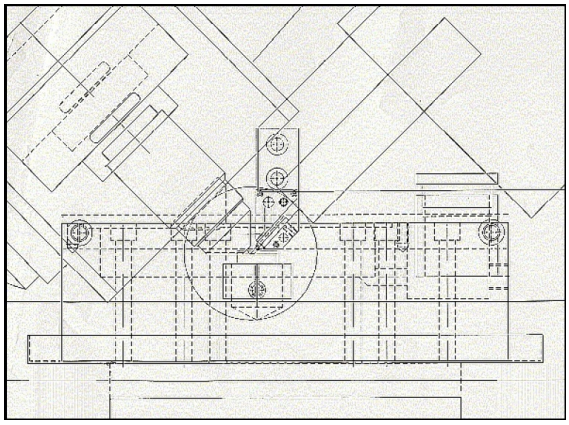
2.1 Design and construction

The KESM was designed by Bruce H. McCormick who was the founding director of the Brain Network Laboratory, and was initially developed as the Brain Tissue Scanner (Fig. 2.1). The basic concept of KESM is the combined scanning with high-resolution microscopy and a diamond microtome. Key features of the KESM include the custom knife-collimator assembly that consists of the white light collimator, and custom diamond microtome; microscope optical train; and high speed line-scan camera (Fig. 2.2).

A white light illumination source provides a $30-100mm$ (the length of the knife-edge) wide strip of intense illumination at the tip of the diamond knife. The custom diamond knife tip is not only used for the physical sectioning of the tissue block but also as an optical element: a prism in the illumination light path. The reflected illumination from the bottom facet of the custom diamond knife proceeds to the objective. The microscope objective focuses on the tip of edge of the knife. The objective and the knife are aligned with each other as their axes are oriented at 45° and 135° to the vertical, respectively. Therefore, the objective is aligned orthogonal



(a) Schematic of precision positioning system, showing granite base and bridge (Aerotech, Inc.)



(b) Specimen undergoing simultaneous sectioning and line scanning. Objective is on left, diamond knife on right. Specimen carrier shown schematically.

Figure 2.1: The Brain Tissue Scanner [25].

to the top surface of knife. The high speed line-scan camera images, through the objective, the illuminated tissue section on the tip of knife during the sectioning process (Fig. 2.3).

Instead of moving the optical elements, a specimen of tissue block on the three-axes air-bearing precision stage is manipulated by a series of mechanical movement in three-axes. The movement employs a stair-step cutting algorithm to obtain the 3D image data set. An air-bearing stage for the three-axes allows ultra-precision positioning that can be adjusted at a $20nm$ resolution for the X- (cutting axis) and Y-axis, and $25nm$ for the vertical Z-axis [25], [26], [21]. The speed of movement of the stage is kept sustainably at $11mm/s$ and the stage position is determined with $20nm$ accuracy.

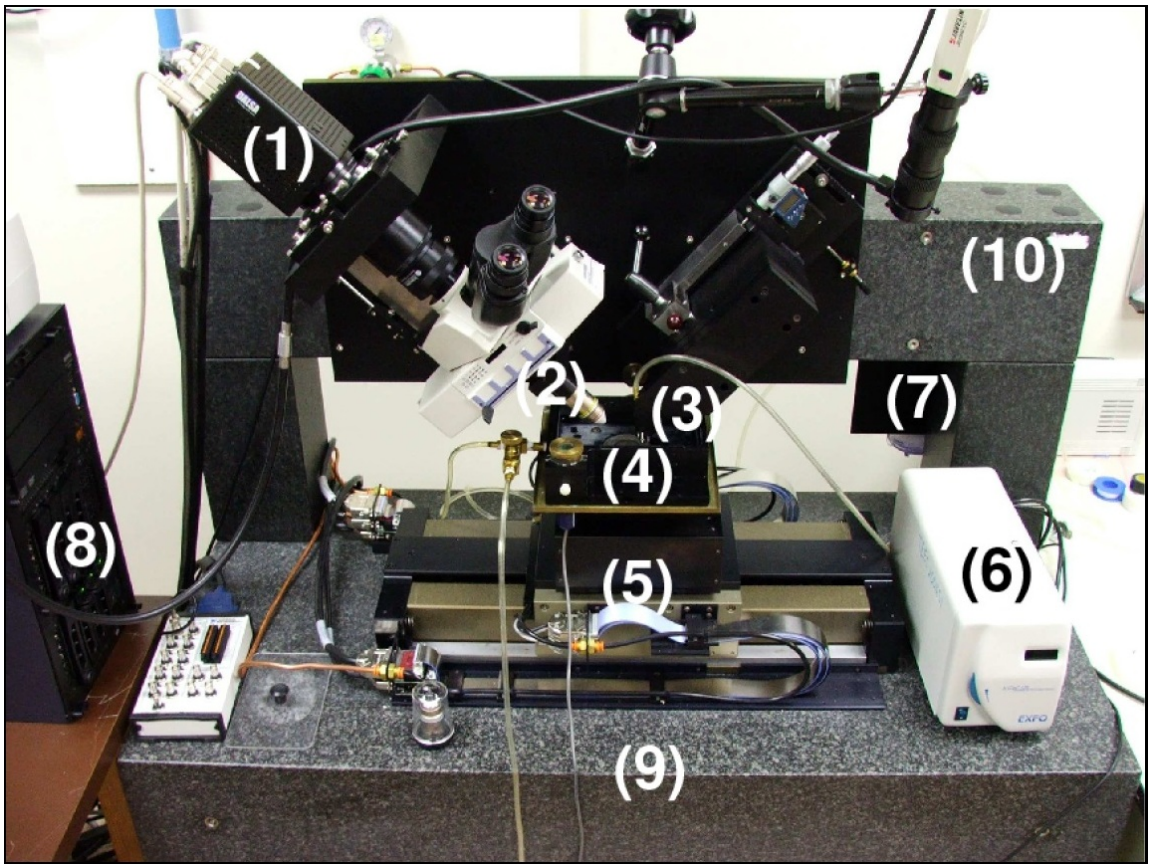


Figure 2.2: The Knife-Edge Scanning Microscope (KESM). A photo of the KESM is shown with its major components marked: (1) high-speed line-scan camera, (2) microscope objective, (3) diamond knife assembly and light collimator, (4) specimen tank (for water immersion imaging), (5) three-axes precision air-bearing 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. Adapted from [4].

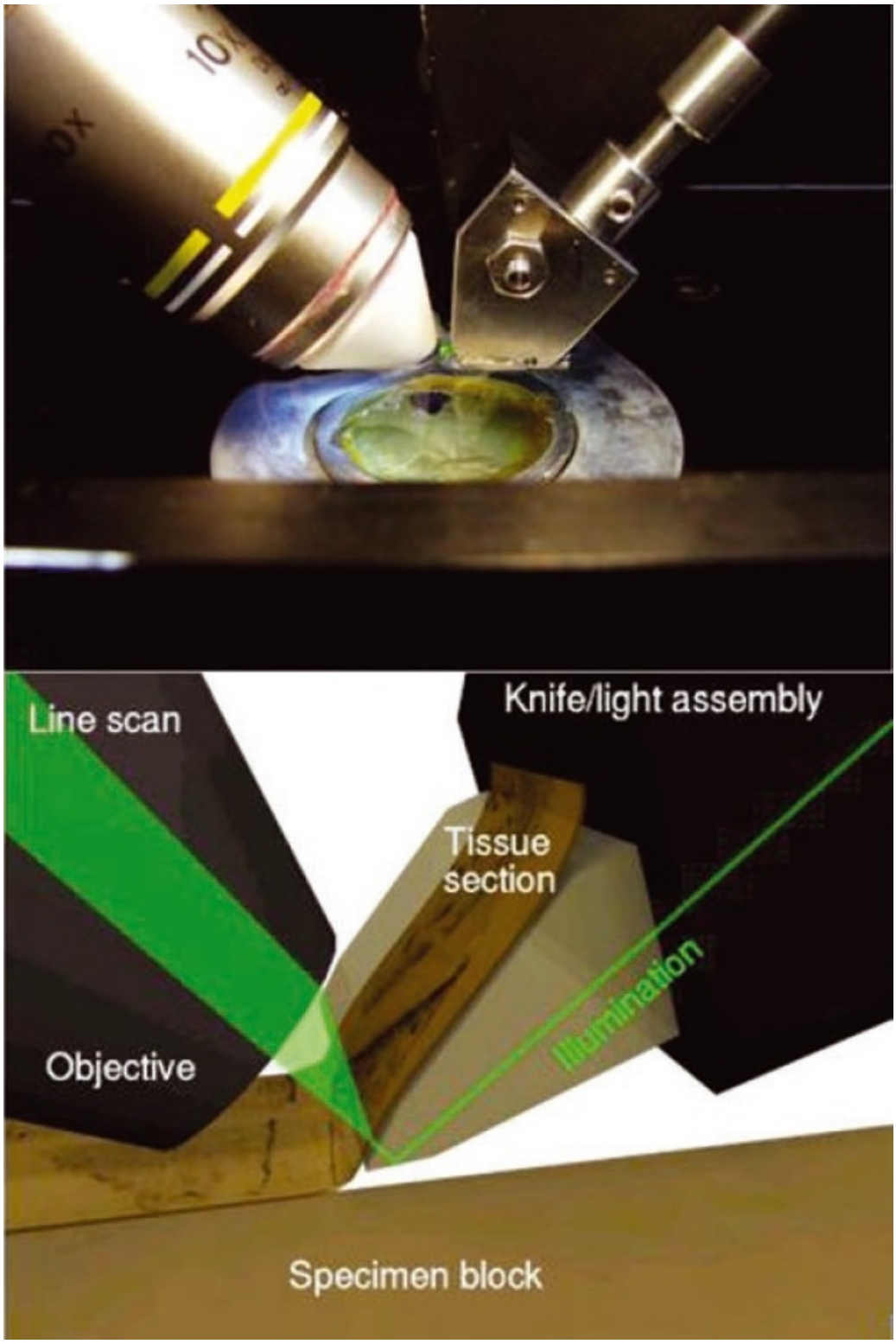


Figure 2.3: Tissue Sectioning and Imaging in KESM. A photogragh of the objective, knife and specimen (top), and a cartoon of the illustrating light transport through the diamond knife and through the objective for imaging (bottom). Adapted from [21].

2.2 Imaging with the KESM

For precision scanning in 3D, a specimen block containing a mouse brain is rigidly mounted on a three-axes precision stage. The KESM uses a custom diamond knife that enables ultra-microtome sectioning of the tissue embedded in plastic as thin as $0.5\mu m$ and is mounted to a massive granite stone to reduce vibration when it is sectioning the plastic block. The three-axes precision air-bearing stage makes the tissue block move against the direction of the edge of custom diamond microtome to section the tissue. Concurrently with sectioning, the newly-cut tissue ribbon on the tip of diamond knife illuminated by the custom knife-collimator assembly is scanned through the microscope objective. However, the imaging width is limited because of the narrow field of view (FOV) of the microscope objective. By automatically controlling the three-axes air-bearing precision stage with the stair-step cutting algorithm (Fig.2.4), we can get the volumetric image data set of the tissue that has minimal damage and dislocation between neighboring columns [17].

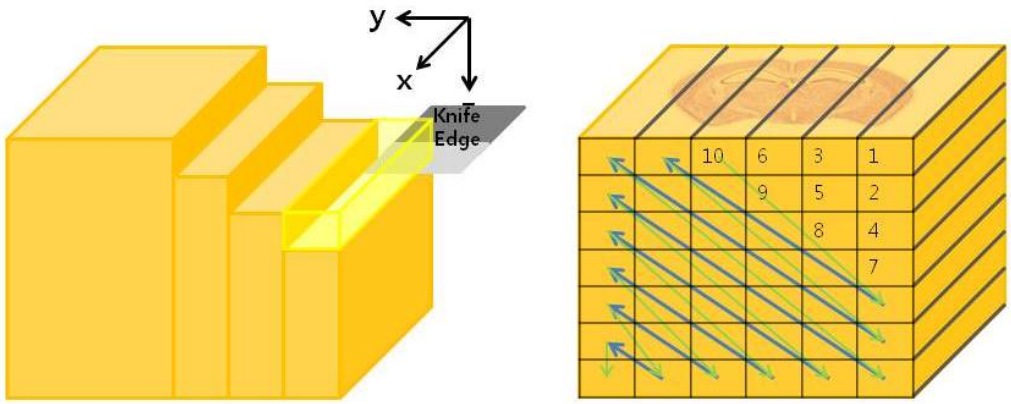


Figure 2.4: Stair-Step Algorithm [17]

Image acquisition is handled by a data acquisition sytem which controls the stage controller and the image capture device (Fig.2.5). The high speed line-scan camera

captures and transmits the high-resolution line images from the objective to the image acquisition card plugged into the computer's PCI-X bus, where finally the images obtained by the high speed line-camera are saved in the data server computer. The system was developed as a custom image capture system that fully automated all the processes [16].

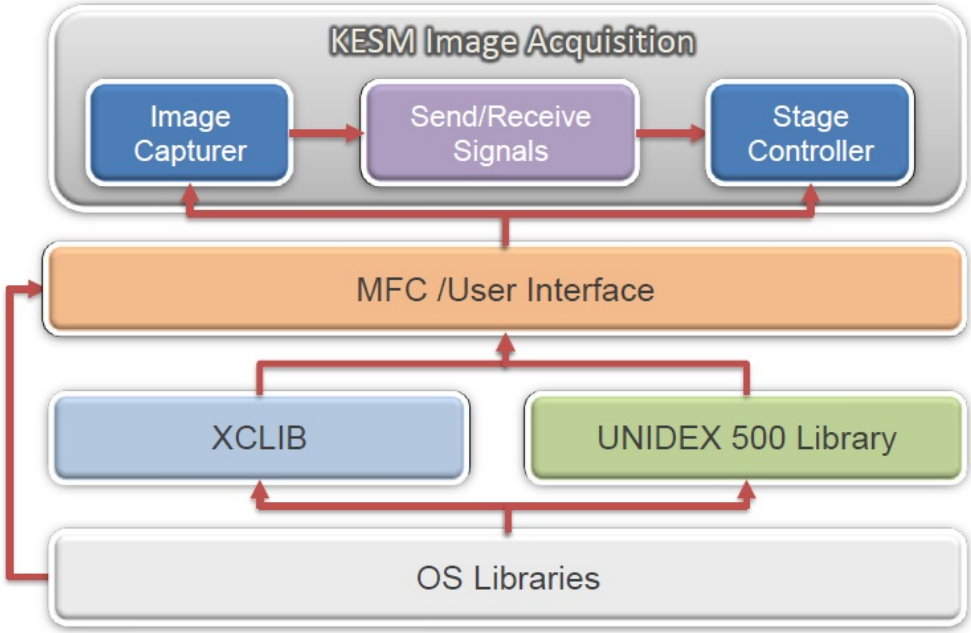


Figure 2.5: Diagram of Data Acquisition System. XCLIB controls the camera actions as a library routine. UNIDEX 500 library controls the precision positioning system. Adapted from [16].

The KESM Image Acquision System (KIAS) includes four components (Fig.2.6): (1) KESM Image Capture System (KICS), (2) KESM Stage Controller (KSC), (3) KESM Session Manager (KSM), (4) KESM Stair-Step Controller (KSSC). The KSC, which is a control software package, controls the movement of the presision positioning system and KSCC according to the Stair-Step algorithm. KSC also triggers KSM to start, stop, pause and resume a cutting session. KICS starts or stops the captur-

ing of images and saves each captured image with a unique name that indicates the positioning parameters. The KSC enables the users to adjust the parameters such as the width, thickness and length of a slice, the cutting speed, and the interval and the duration of the triggering signal for imaging.

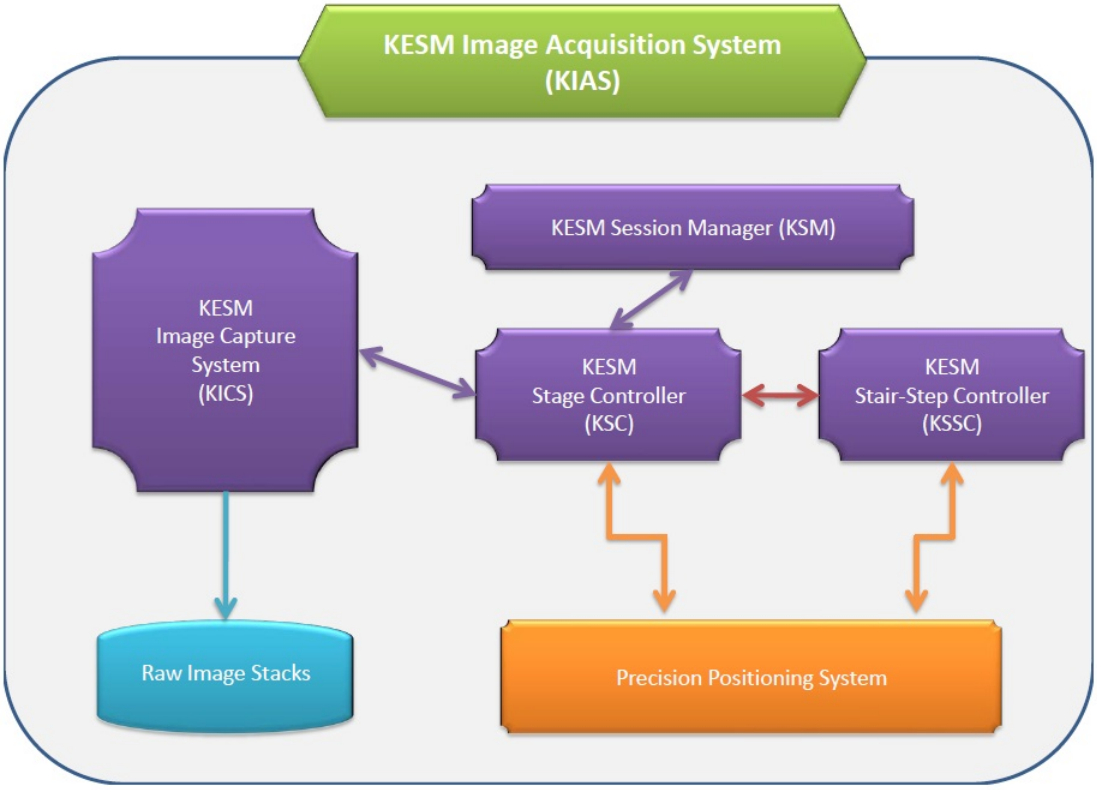
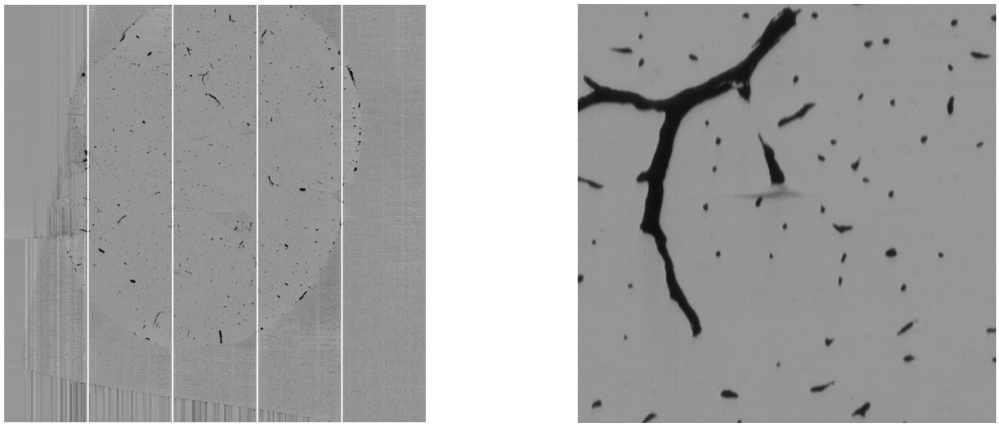


Figure 2.6: An overview of KESM Image Acquision System (KIAS) including four components and their relation. (1) KESM Image Capture System (KICS), captures images through the high speed camera. (2) KESM Stage Controller (KSC), controls the overall KIAS operations. (3) KESM Session Manager (KSM) manages cutting sessions to ensure the continuity from all information of the previous status. (4) KESM Stair-Step Controller, supports the KSC to move along the Stair-Step algorithm. Adapted from [16].

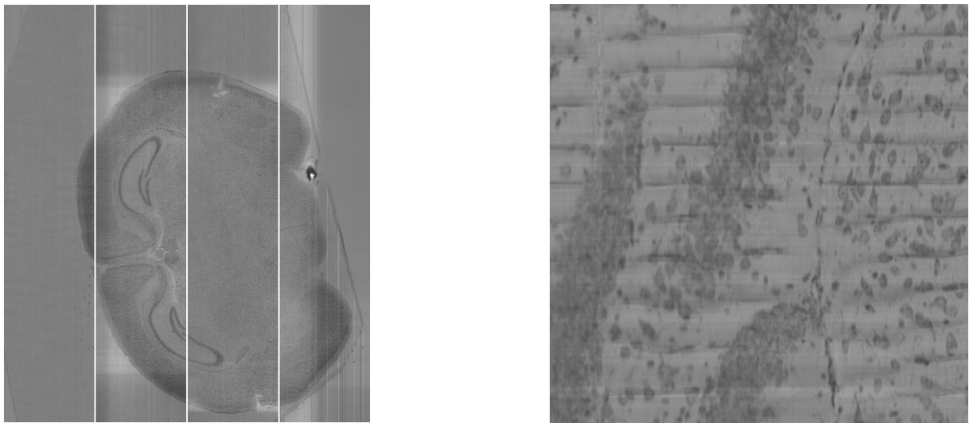
The KESM has a capability to scan images of biological tissue (about $1cm^3$) at a $0.6\mu m \times 0.7\mu m \times 1.0\mu m$ resolution within 100 hours, and to generate data at 180 Megabytes/second (MB/s).

2.3 KESM Data sets

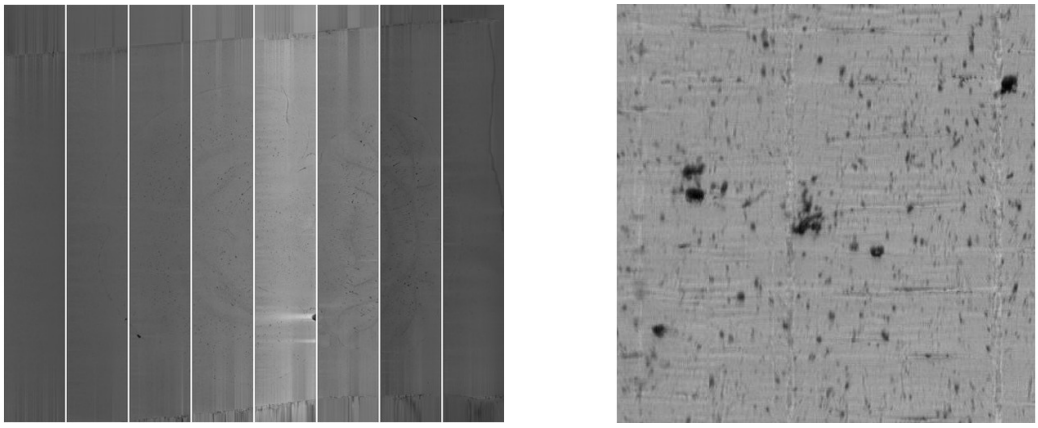
We obtained three kinds of stained whole mouse brain image data from KESM (Fig.2.7). The first whole-brain-scale data set we obtained was Golgi-stained and India ink-stained mouse (genotype C57BL/6J) brains in 2008 [6], [1]. Golgi-Cox staining was used for the neuronal morphology visualization of the mouse brain (Fig.2.8, 2.11, 2.12) and India ink staining was used to reveal the vascular networks of the mouse brain (Fig.2.9, 2.13). In early 2010, we also scanned a whole mouse brain stained with Nissl that was used to map the soma distribution over the entire mouse brain [5] (Fig.2.10). Finally, another Golgi-stained brain was fully imaged later in 2010 [7], [8].



(a) Tissue section columns of KESM India ink data (left) and a close-up (right).



(b) Tissue section columns of KESM Nissl data (left) and a close-up (right).



(c) Tissue section columns of KESM Golgi data (left) and a close-up (right).

Figure 2.7: Three kinds of KESM data (a) India ink data showing vascular networks, (b) Nissl data showing soma distribution, and (c) Golgi data showing neuronal morphology.

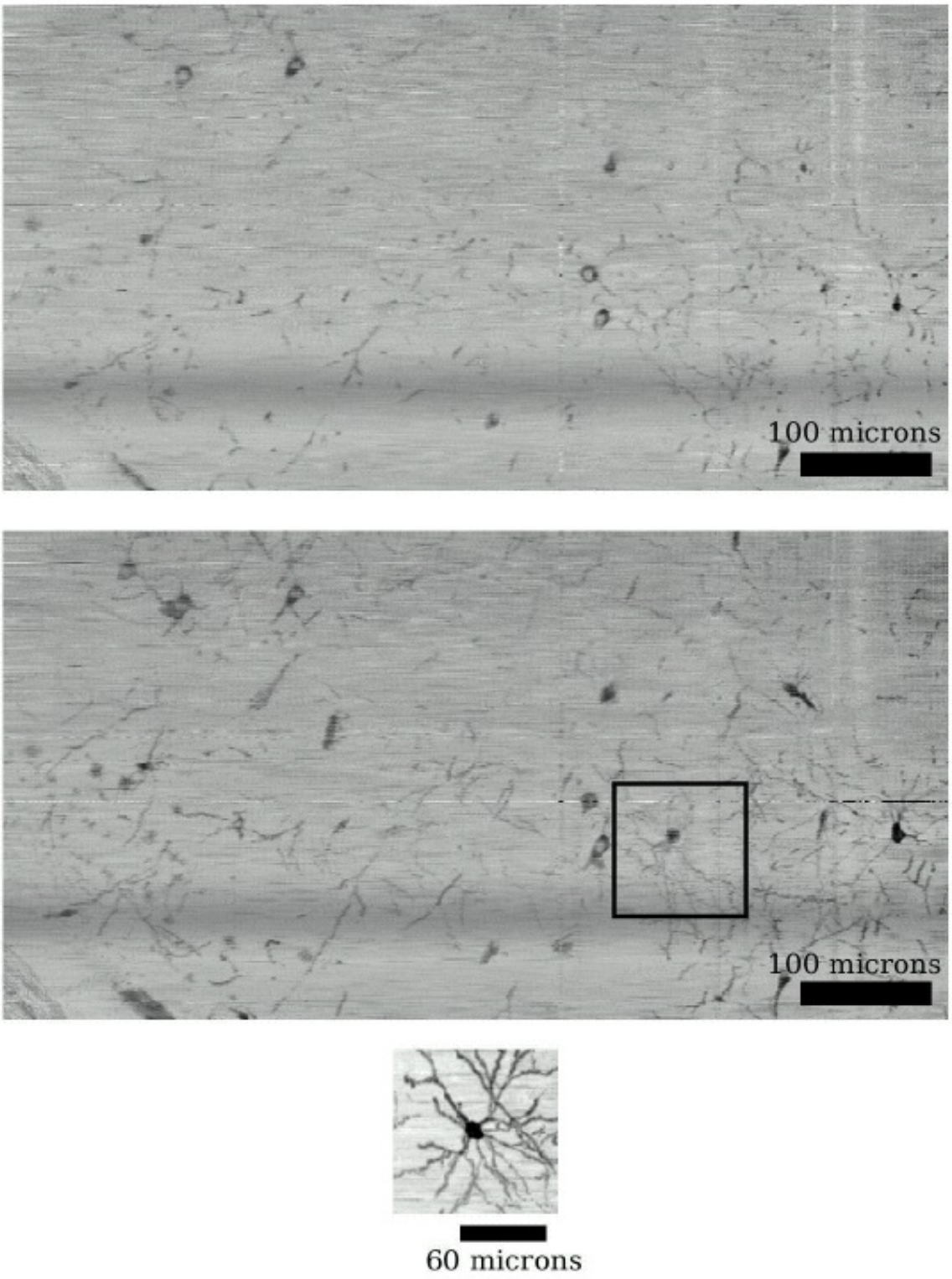


Figure 2.8: Sagittal section (resectioning) of the KESM Golgi data. A single resectioned slice (top), an overlaid series of images (middle), and a single neuron shown in an overlaid stack of 300 images (bottom). Adapted from [21].

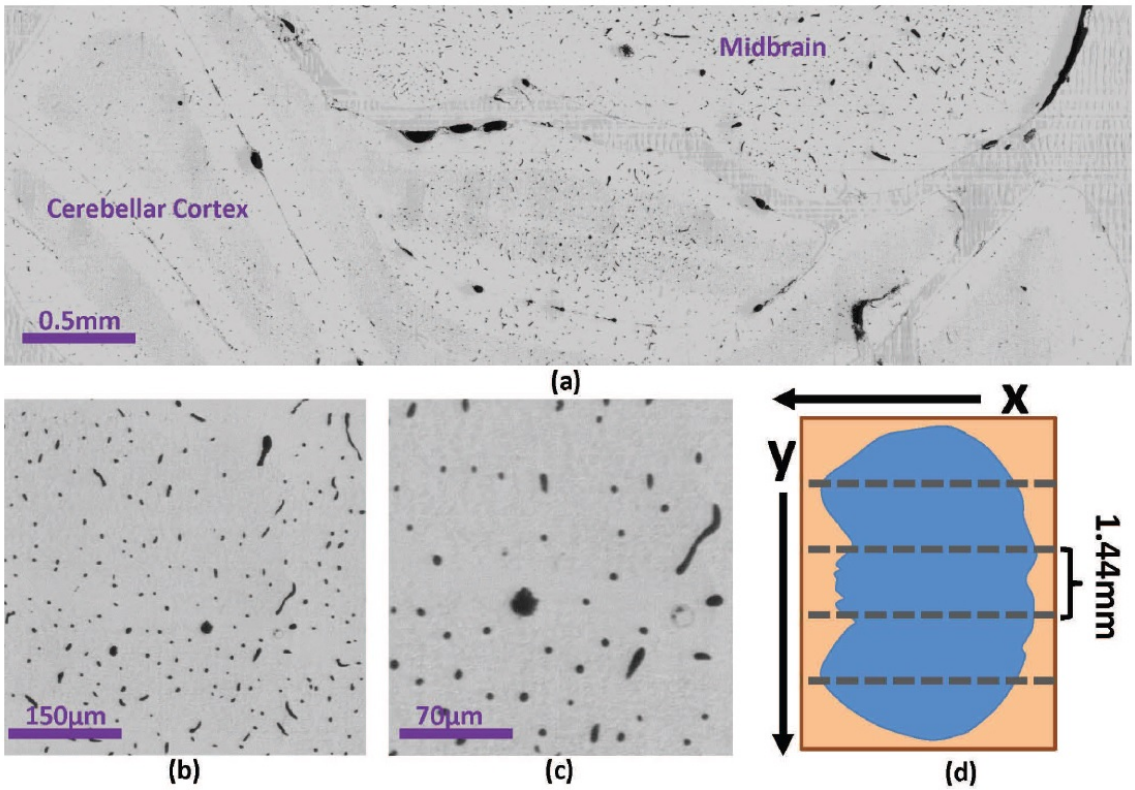


Figure 2.9: Images from mouse cerebellum and mid-brain stained with India ink showing vasculatures. (a) Single coronal KESM section and closeup showing a (b) $500\mu m$ and (c) $200\mu m$ view of the same section. (d) A single tissue cross-section consists of several columns due to the limited field of view of the objective. Adapted from [22].

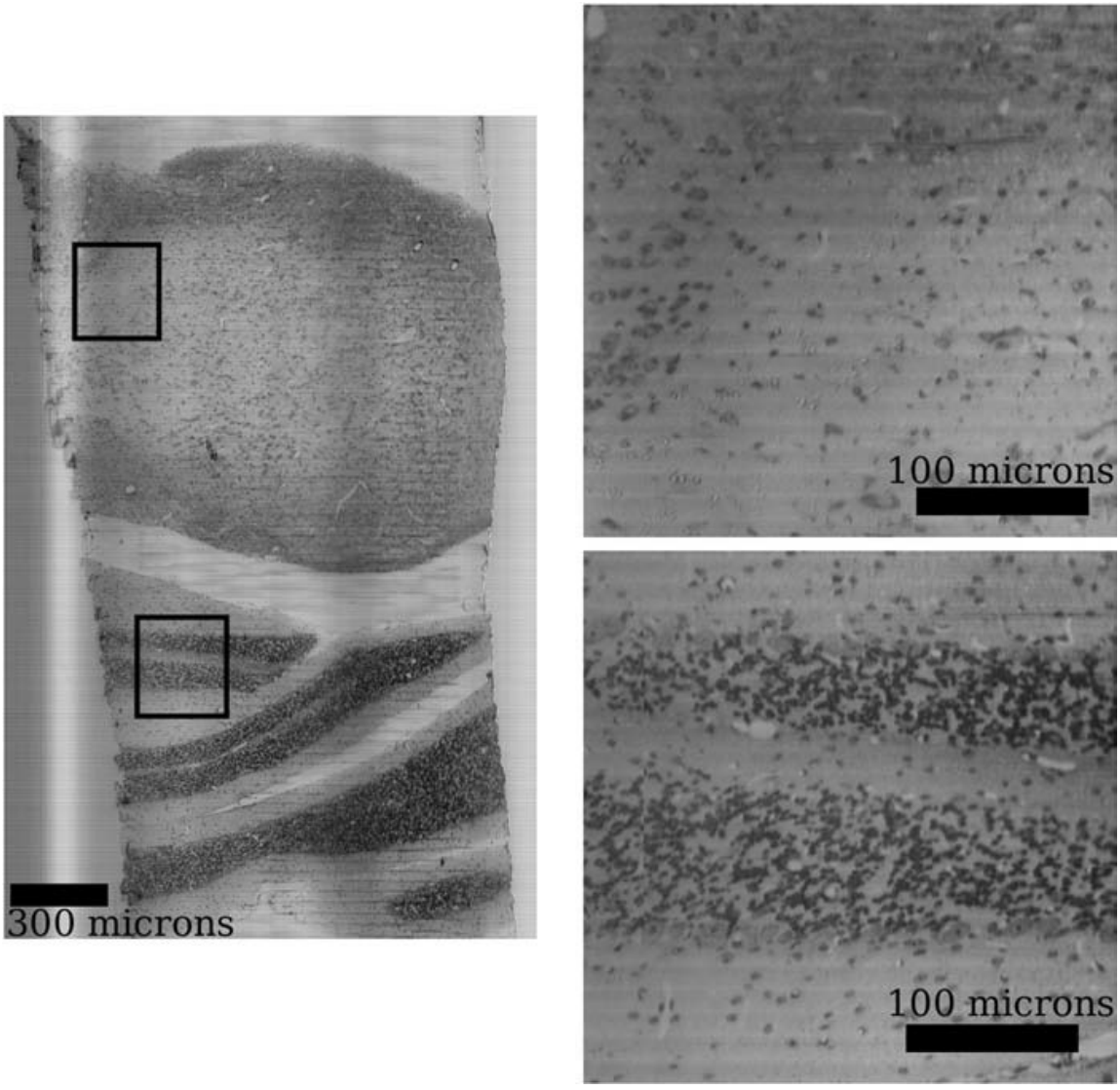


Figure 2.10: Coronal section of mouse brain stem and cerebellum (KESM Nissl data). The complete section (left), and close-ups (right). The resolution is $0.6\mu m \times 0.7\mu m$, and each section corresponds to a $1\mu m$-thin tissue section. Adapted from [21].

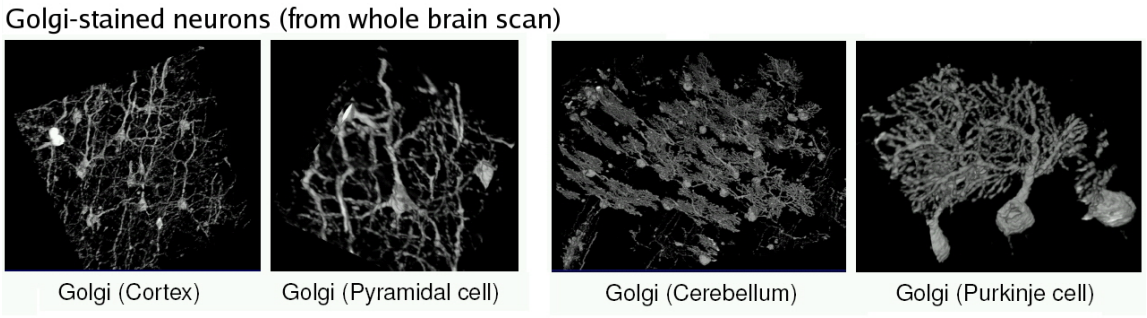
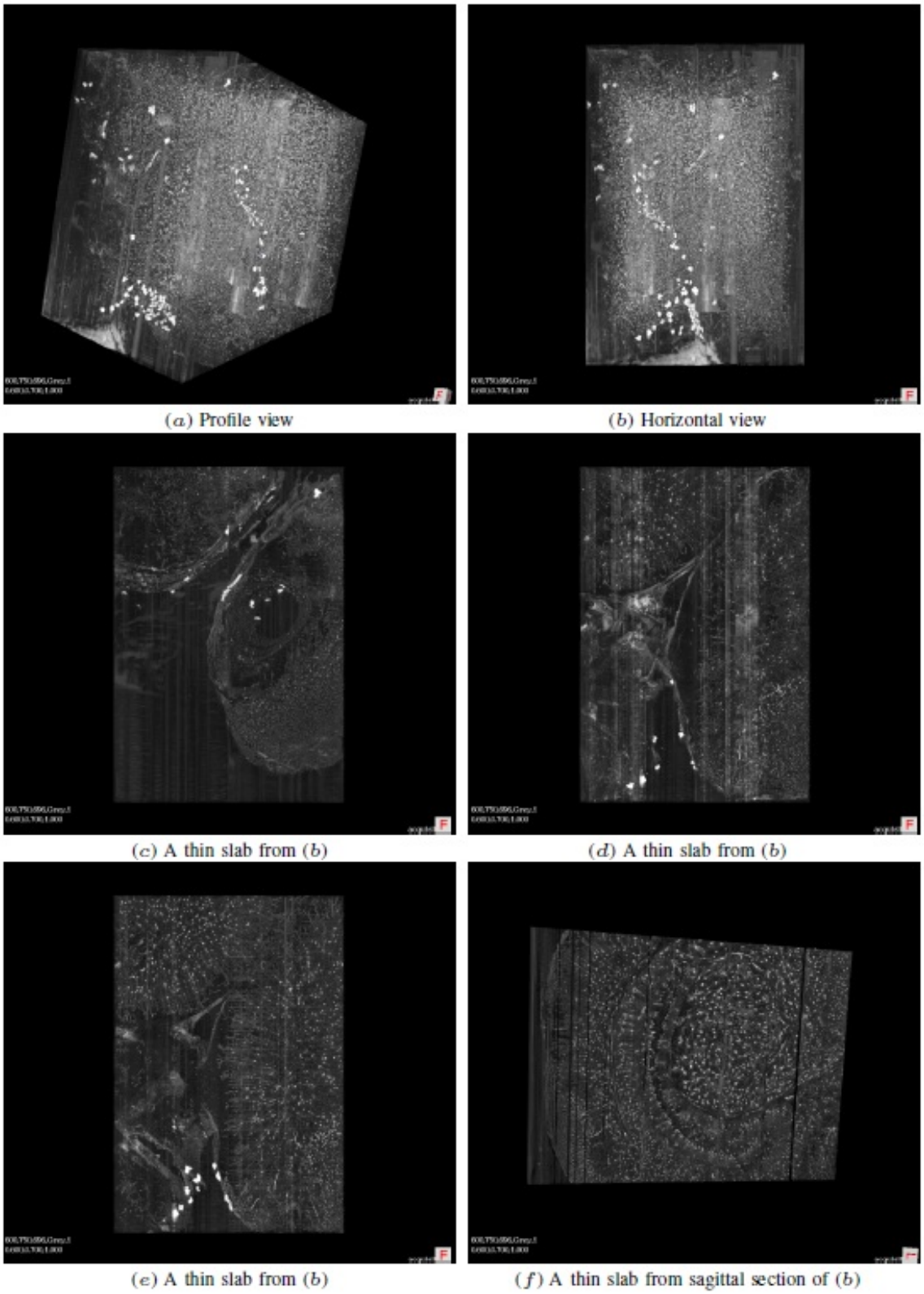


Figure 2.11: 3D visualization of the KESM Golgi data. Neurons from different regions in the KESM Golgi data set. The first figure is the cortex region ($\sim 300\mu m$), and second is a close-up of the first ($\sim 150\mu m$). The third figure is the cerebellum region ($\sim 500\mu m$). The last figure shows Purkinje cells ($\sim 100\mu m$). Adapted from [6].



(a) Profile view

(b) Horizontal view

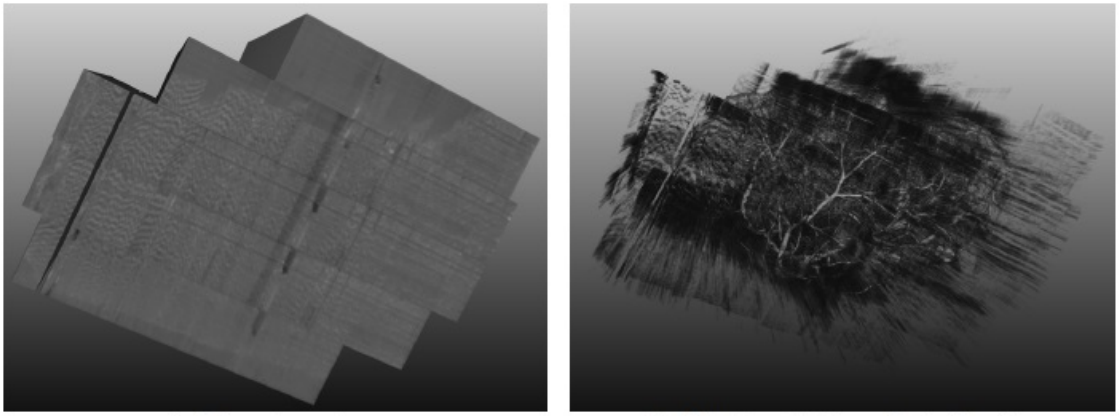
(c) A thin slab from (b)

(d) A thin slab from (b)

(e) A thin slab from (b)

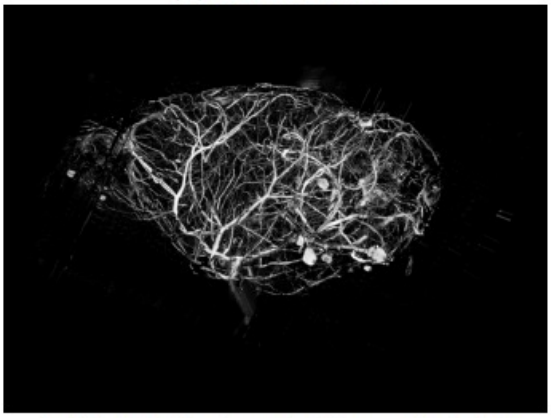
(f) A thin slab from sagittal section of (b)

Figure 2.12: Different views in 3D of the KESM Golgi data. The Block width is a $2.88mm$. Adapted from [6].

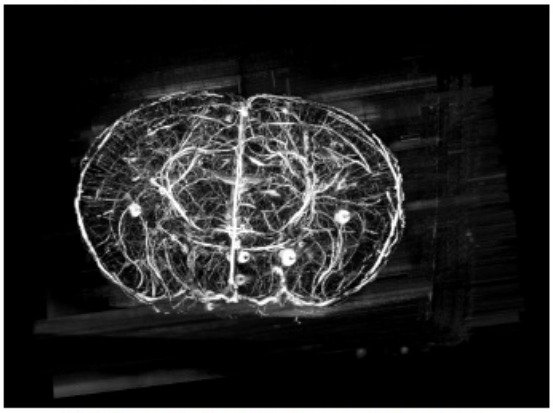


(a) Raw data volume

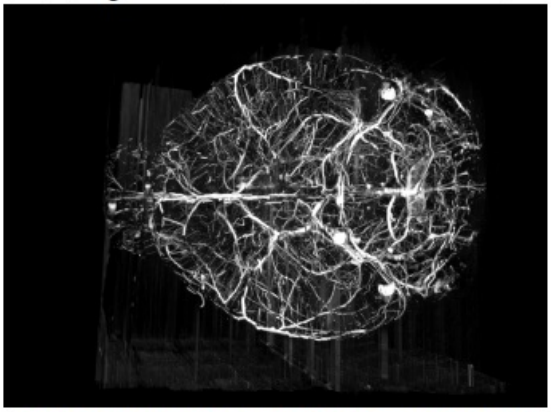
(b) Initial thresholding of (a). See (c).



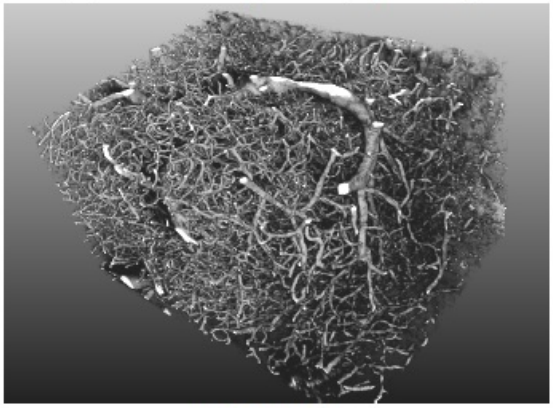
(c) Sagittal view (←: Anterior, →: Posterior)



(d) Coronal view (↑: Dorsal, ↓: Ventral)



(e) Horizontal view (←: Anterior, →: Posterior)



(f) Close-up

Figure 2.13: Visualization of the vascular networks in the KESM India ink data. (a) 3D view of raw data in the sagittal plane. (b) A lightly thresholded version of (a). (c)−(e) Fully thresholded versions of (a) but in a different view ($\sim 5mm$). (f) Close-up of the intricate details ($\sim 1.5mm$). Adapted from [6], [1].

3. RELATED WORK

Advances in high-throughput, high-resolution microscopy technology can produce hugh amounts of data from biological specimen. A digital atlas is a suitable tool to analyze and share the massive volumetric data from such microscopy instruments. Therefore numerous brain atlases exist to explore, annotate, and share the data sets, and the data modality and approach of each atlas are diverse. Among them, we compared representative brain atlases in neuroscience.

The Allen Brain Atlas (ABA; http://www.brain-map.org/), published by the Allen Institute for Brain Science, provides seven kinds of brain atlases that include mouse brain atlases, a web-based viewing application to analyze genome-wide image data from of the mouse brain. They collected the expression of more than 20,000 genes using *in situ* hybridization (ISH) data from adult C57BL/6J mouse brain at a resolution of $0.95\mu m \times 0.95\mu m \times 25\mu m$. Although ABA supports 5 zoom levels, it cannot guarantee z-axis navigation function, which ensures location connectivity between consecutive images. They also provide a 3D viewer desktop application, the Brain Explorer, which enables the visualization of the ABA gene expression data in 3D at $100\mu m^3$ resolution, but it needs to be installed separately [19], [18], [15], [35], Fig.3.1(a).

Supporting the concept of "virtual microscopy", BrainMaps.org (http://brainmaps.org/) was launched with a quick 2D image viewer with high-resolution ($0.46\mu m$/ pixel, but the thickness is $25 - 30\mu m$) whole-brain images. To support web-accessible images, they employed quad-tree image pyramid tiles for multi-resolution, high-resolution digital brain images and implemented the graphical user interface (GUI) using Flash, Java and AJAX (dHTML and JavaScript). As a result, they provide rapid web-based

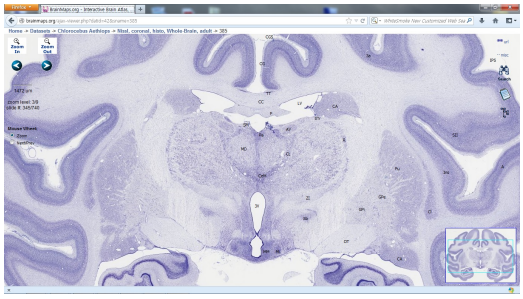
navigation and visualization, aiming to provide a virtual microscopy service. They also provide 3D visualization using OpenGL as a separate desktop application, StackVis, which fills the gap between the web-based 2D image viewer and a full-blown 3D volume rendering system. They took an alternative approach to provide 3D effect via perspective and orthogrphic viewpoint using transparent background image instead of 3D volume rendering since each image in the data set was fairly thick ($25-30\mu m$) [28], [27], [36], Fig.3.1(b).

The Collaborative Annotation Toolkit for Massive Amounts of Image Data (CATMAID; http://catmaid.org/) was inspired by Google Maps, but they seek a decentralized web interface atlas using the web browser. Because of that, they provide downloadable scripts instead of a website for the use of the atlas. The scripts are implemented using JavaScript and PHP and third party plugins are not required (execpt for ImageMagick, a stand-alone image processing application). It enables navigation and sharing of annotations of massive image data sets from serial section Transmission Electron Microscopy (ssTEM) at $4nm \times 4nm \times 60nm$ resolution. They provide collaborative annotation by storing meta-information about data sets in user databases in the CATMAID data server [31], Fig. 3.1(c).

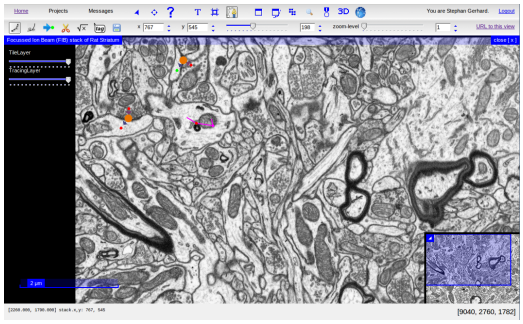
The Edinburgh Mouse Atlas (EMAP; http://www.emouseatlas.org/) is an atlas for the detailed spatio-temporal data of the developing mouse embryo. In contrast to other brain atlases, EMAP provides time-series data that show full mouse embryos in 3D throughout the developmental stages at a medium-resolution of $2\mu m \times 2\mu m \times 2\mu m$ or $4\mu m \times 4\mu m \times 7\mu m$. They provide the online-accessible 2D and 3D atlases using Java applications, but the 3D atlas is more of a cross sectional image viewer instead of a 3D rendering viewer [3], [38], Fig.3.1(d).

Based on the analysis of the characteristics of these digital brain atlases (Table 3.1), we designed the KESMBA to retain the strengths and address the weaknesses.

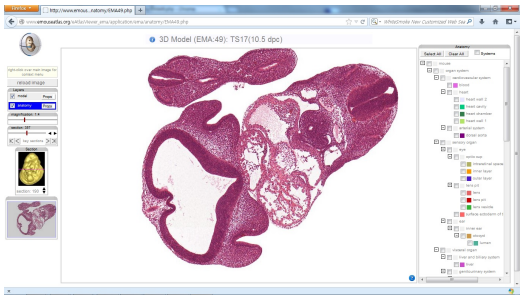
(a) ABA. Source: http://mouse.brain-map.org/experiment/show/73931410/.



(b) BrainMaps.org. Source: http://brainmaps.org/ajax-viewer.php?datid=42&sname=385/.



(c) CATMAID. Source: http://catmaid.org/screenshots.html/.



(d) EMAP. Source: http://www.emouseatlas.org/eAtlasViewer_ema/application/ema/anatomy/EMA49.php/.

Figure 3.1: The screenshots of representative digital brain atlases.

	Species	resolution	2D atlas	Navigation/ Multiscale	3D viewer	description
ABA	Mouse (C57BL/6J), Human	0.95μm × 0.95μm × 25μm	Web-based application	No/ 5 zoom level	Brain Explorer ($100\mu m^3$)	ISH data across more than 20,000 genes can be compared.
BrainMaps.org	7 kinds of species including Homo sapiens.	0.46μm × 0.46μm × 22-30μm	Web-based application (virtual microscopy)	Yes/ 9 zoom level	StackVis (perspective shot)	StackVis fills the gap between 2D image viewer and 3D volume rendering system, using transparent back ground image.
CATMAID	Mouse V1 Mouse S1 C.elegans	4 nm × 4 nm × 60 nm	Web-interface Script	Yes/ ~ 5 zoom level	N/A	collaborative annotation by storing meta-information about datasets.
EMAP	Time-series of mouse-embryo	2μm × 2μm × 2μm or 4 μm × 4 μm × 7 μm	Web-based Atlas using Java Webstart	No/ 4 zoom level	Java Atlas Viewer (Cross section from any orientation)	Provided a time-series of 3D mouse embryo by mapping process which is a simple linear affine transform (scale, rotate, translate)

Table 3.1: Comparison of characteristics of the representative digital atlases.

4. PRIOR WORK: KESM BRAIN ATLAS

As previously stated in Chaper 2, our KESM data sets are large in volume (several tera voxels) and high-resolution. It is a great challenge to distribute the data and to visualize the whole brain image for analysis due to the large volume and high-resolution. Therefore, we developed a web-based brain atlas using the Google Maps API, the KESM Brain Atlas (KESMBA) [2].

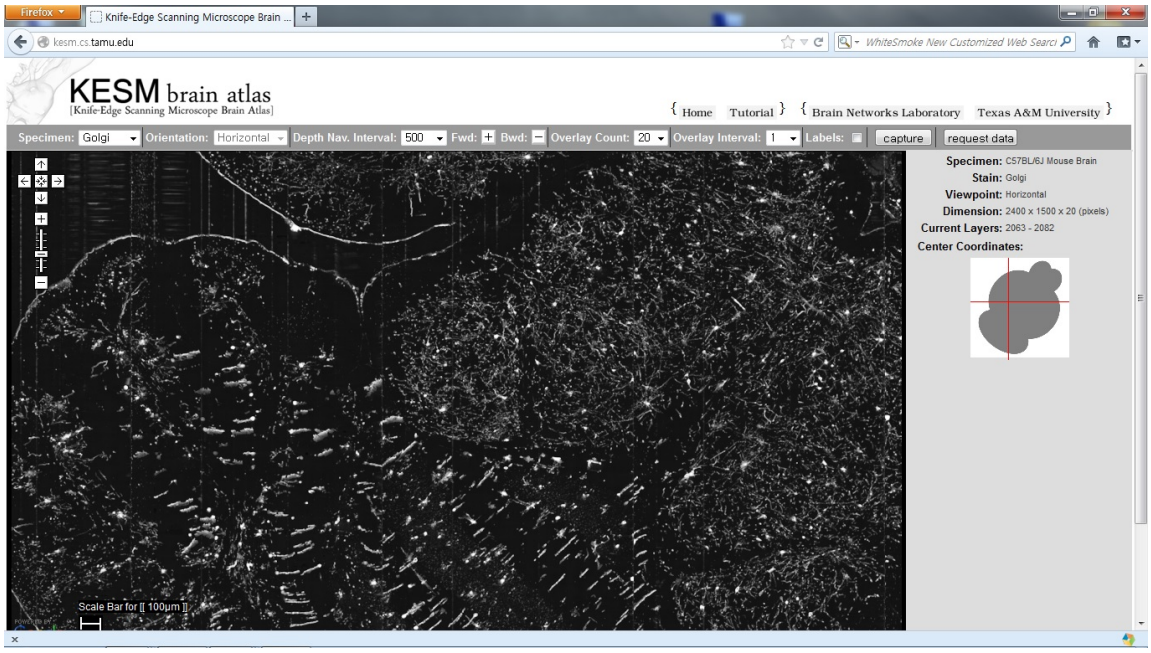


Figure 4.1: KESM Brain Atlas. Source: http://kesm.cs.tamu.edu/.

4.1 Basic Concept of KESMBA

For extensive dissemination of KESM on the web, we designed the KESMBA using Google Maps API, a popular geographic mapping framework. To understand the structure of the brain, three dimensional visualization is mandatory. However, it is not possible to visualize a nearly 2 TB data on a typical desktop computer, even

when it is equipped with a high-end graphics card. Thus we decided to find a way to visualize the data sets in 3D, for the widest possible dissemination using only a standard web browser on a machine, without the requirement of high-performance graphics devices and add-on applications. To meet these specifications, we employed distance attenuation by overlaying Portable Network Graphics (PNG) images with the background made transparent, all within the Google Maps API. Finally, we employed pyramidal multiscale tiling to provide multiscale visualization from the cellular level to the entire brain level.

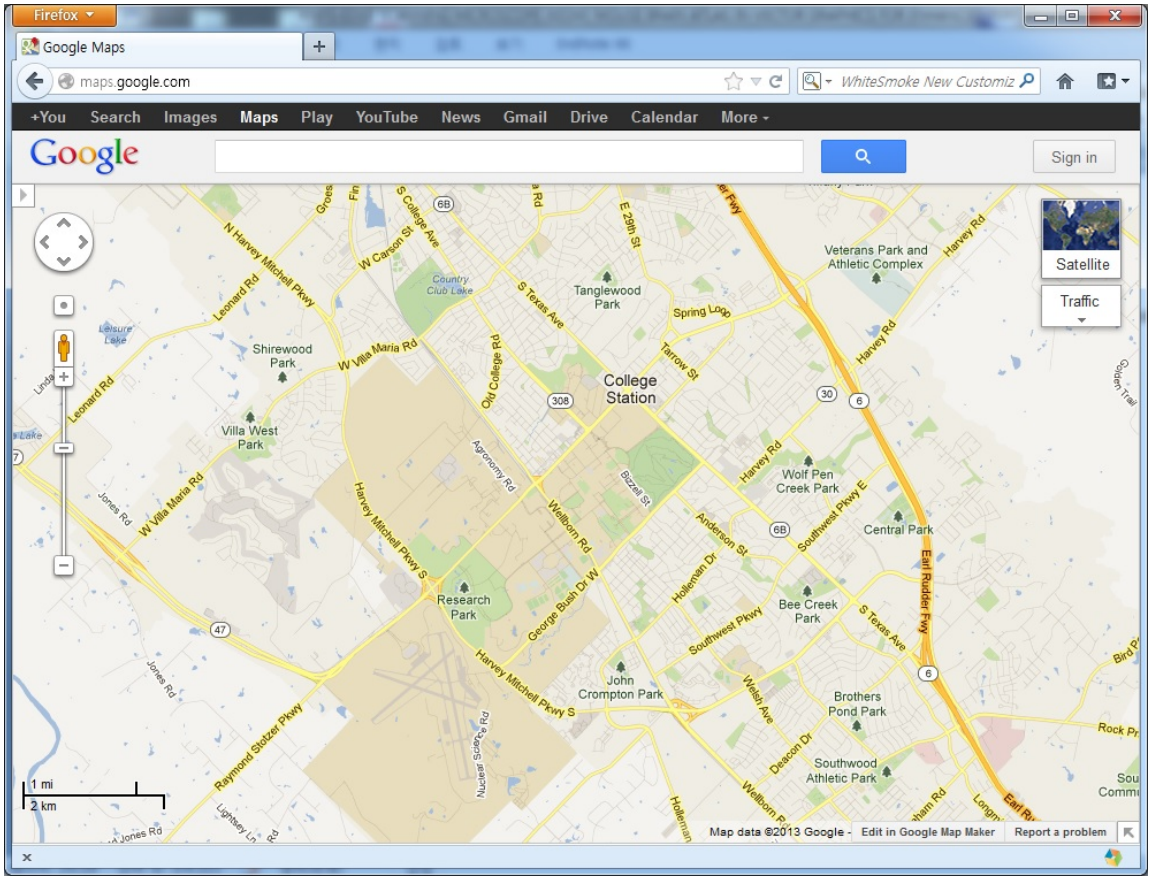


Figure 4.2: Google Maps web mapping service. Source: http://maps.google.com/.

4.2 Google Maps API

Google provides the geometric mapping service on the web, Google Maps (Fig.4.2), and they also allow developers to employ the Google Maps API that can be customized into their websites using the users' own map data [11]. We designed the KESMBA using the Google Maps JavaScript API V2 that provides the key functions of navigation and overlay, and customized the API for the KESMBA: custom tiles, tile overlay, user options to select the number and interval of the overlay, overlaying zoomable annotation, and map redraw function. On top of that, we added new features such as an information panel, scale bar, map capture button, and z-axis navigation controller (Fig.4.3).

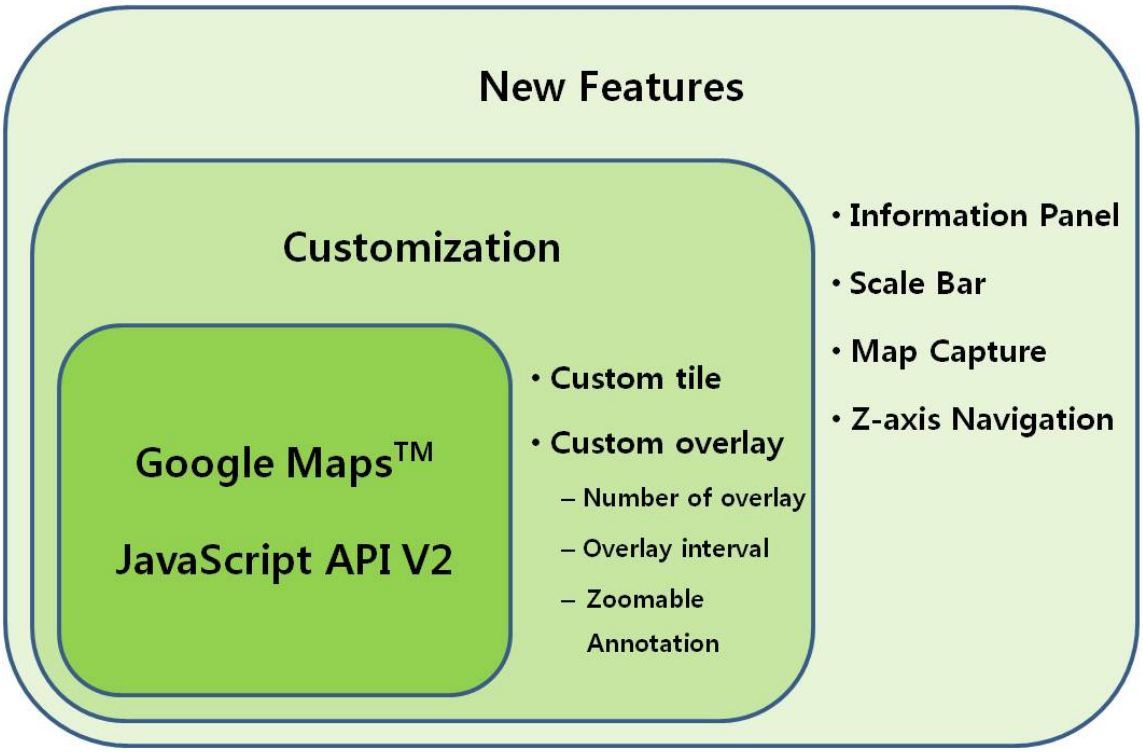


Figure 4.3: Structure of the KESMBA using Google Maps API.

We implemented the user interface for the KESMBA using PHP, JavaScript, and Cascading Style Sheets (CSS). In the main page with PHP, we loaded Google Maps JavaScript API and employed the basic fuction of zooming and panning provided by the API, added Javascript code to customize the KESMBA, and provided a menu bar for selecting different options. The Google Maps JavaScript API is imported from Google using a license key. Key parts of the main page (index.php) and the overlay function (overlay.js) are shown below.

```
//////////////////////////// File: index.php ////////////////////////////
<script src="http://maps.google.com/maps?file=api&amp;v=2&amp;
sensor=false&amp; key=AIzaSyCZ_Vyj5u2ildB0PHeR8RY-ungoyrmI2s4"
type="text/javascript"></script>
<script src="./scripts/overlay.js" type="text/javascript"></script>

// Menu bar
<div class="hor_menu">
<table class="upperMenu">
...
// 1. Data selection
<th>Specimen:</th>
<select id="specimen" name="specimen" onchange="newSpecimen(this);">
...
// 2. Orientation selection
<th>Orientation:</th>
<select id="plane" name="plane" disabled="disabled" onchange="newPlane();">
...
```

```
...
// 3. Z-axis Navigation
<th>Depth Nav. Interval:</th>
<select id="depthMargin" name="depthMargin" onchange="newDepthMargin(this);">
<img class="btnImg" id="mvIn" name="mvIn" src="./images/plus1.png"
alt="Move-in(deeper)" onmousedown="pressPlus();" onmouseup="releasePlus();"
onclick="moveIn();">
<img class="btnImg" id="mvOut" name="mvOut" src="./images/minus1.png"
alt="Move-out(shallower)" onmousedown="pressMinus();" onmouseup="releaseMinus();"
onclick="moveOut();">
...
// 4. Number of overlay
<th>Overlay Count:</th>
<select id="numOverlay" name="numOverlay" onchange="newNumOverlay(this);">
...
// 5. Overlay interval
<th>Overlay Interval:</th>
<select id="intOverlay" name="intOverlay" onchange="newIntOverlay(this);">
...
// 6. Zoomable Annotation
<th>Labels:</th>
<input type="checkbox" id="checkLabels" unchecked onclick="clickedLabelCheckBox();" />
...
// 7. Map capture
<td class="border-style"><input id="print" name="print" type="button"
style="height:22px;width:70px;" value="capture" onclick="window.open('print.html');" />
```

```
//////////////////////////// File:overlay.js ////////////////////////////
...
// 1.Createatilelayer.
customLayer=[new GTileLayer(...)];
...
// 2.GenerateacustomtileURL.
customLayer.getTileUrl=customGetTileUrl;
...
// 3.Createanoverlayinstanceof
// theGTileLayerOverlayclass.
customOverlay=new GTileLayerOverlay
(customLayer);
...
// 4.Createamaptypeinstanceofthe
// GMapTypeclass.
customMap=new GMapType(...);
...
// 5.CreateamapinstanceoftheGMap2 class.
var myMap=newGMap2(document.getElementById("map"),
mapTypes:[customMap]);
...
// 6.Addpredefinedcustommaptypeinto themapinstance.
myMap.addMapType(customMap);
...
// 7.Addacustomtilelayerintothemap instance.
cMap.addOverlay(overlays);
```

4.3 Distance attenuation

The basic idea of distance attenuation using transparent image stack for KESM 3D data rendering was proposed by Eng and Choe [10]. Distance attenuation is inspired by the haze effect in natural landscapes (Fig.4.4).

Figure 4.4: Haze effect in natural landscape. Objects in the foreground appear darker and high contrast compared to those in the background. Photograph courtesy of Yoonsuck Choe (2010).

When we see a hazy landscape as in Fig.4.4, closer objects appear to have high contrast and farther objects low contrast. Distance attenuation can be shown more dramatically with a series of synthetic images (Fig.4.5). In the figure, we can see a series of images with the same intensity that make up two intertwining branches (Fig.4.5A). When we overlay all of the images using Minimum Intensity Projection

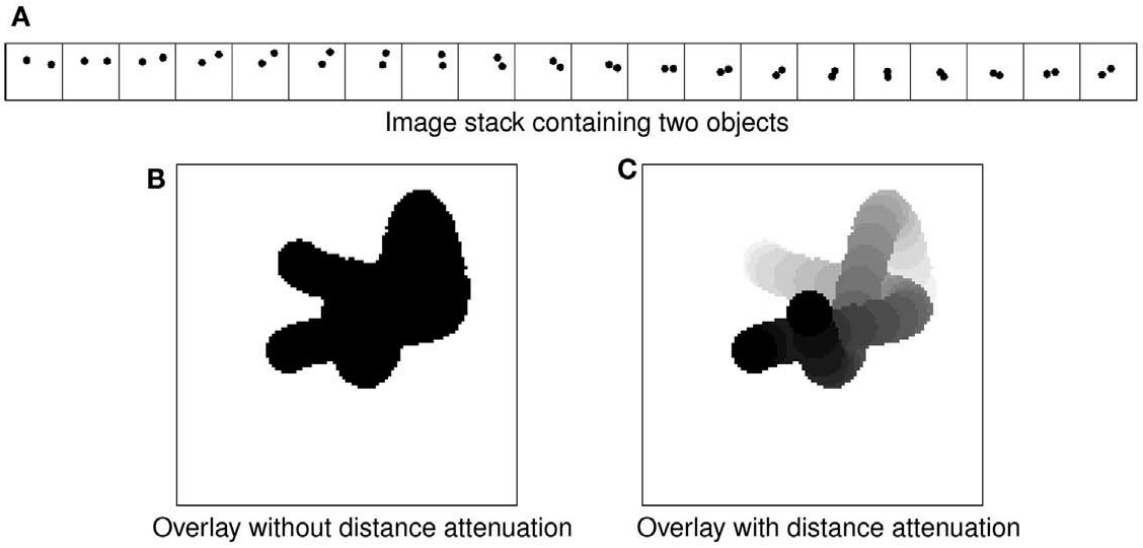


Figure 4.5: Transparent overlay with distance attenuation. (A) An image stack containing two intertwined objects are shown. (B) Simple overlay of the image stack in (A) results in loss of 3D perspective. (C) Overlay with distance attenuation helps bring out the 3D cue. Adapted from [10].

(MIP) without distance attenuation, it is hard to figure out what are the embedded objects (Fig.4.5B). We applied distance attenuation by increasing intensity from 0.05 to 1.0 to each of the twenty images in the stack and overlaid them (Fig.4.5C). From the overlaid image with distance attenuation, we can easily recognize the two intertwining objects, recovering 3D information from the 2D images.

To apply this technique to the KESM data, we need preprocessing of the acquired raw images. The first step is to invert the intensity of the image. After inverting, we employed Gamma correction with a sigmoidal non-linearity to enhance the contrast between the foreground objects and the background. The last step is to make the background transparent in order to overlay the images with distance attenuation. The final images are stored as Portable Network Graphics (PNG) format with added alpha channel for transparency.

4.4 Multiscale Zoom level

When neuroscientists are analyzing the brain microstructure and connectome at the cellular level, they would not only need high-resolution images from small regions of interest but also a map of the entire brain to recognize where those regions are located in the brain. Therefore our KESMBA should provide multiscale image pyramids. After preprocessing and adding transparency, we generated multiscale tile pyramids (Fig.4.6A). We already have high-resolution images, thus we prepared different zoom level tiles from the bottom-up to represent from the cellular level to the full brain level.

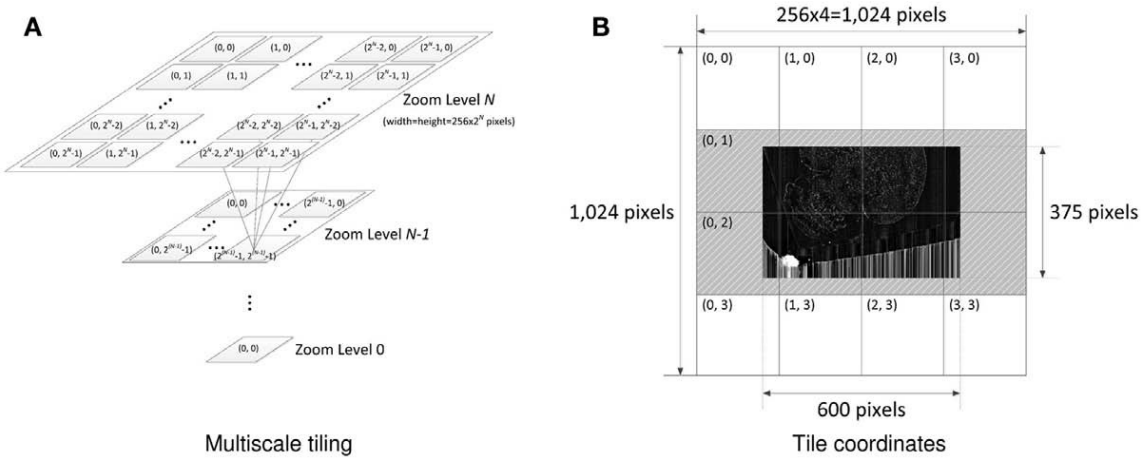


Figure 4.6: Multiscale tiling scheme. (A) Zoom level N is composed of $2^N \times 2^N$ tiles with each tile being 256×256 pixels. (B) Example of KESM Golgi tiles in zoom level 2, where the entire brain can be displayed in a single screen. Adapted from [8].

Our Golgi data set consists of 8 columns where each column has thousands of images that are 2,400 by 12,000 pixels in size. Therefore the size of one layer of the KESM Golgi images is 19,200 by 12,000 pixels and we need to generate at least 75×47 tiles since the Google Maps API supports tiles with 256×256 pixels. The number of tiles of each zoom level is $2^N \times 2^N$ (N represents the zoom level). For

example, level 7 can be represented fully with 128×128 tiles at our original KESM resolution. From level 7, we scale down and generate the tiles of each higher level step by step to level 2, where the entire brain of the KESM Golgi data set can be displayed in a single screen (Fig.4.6B).

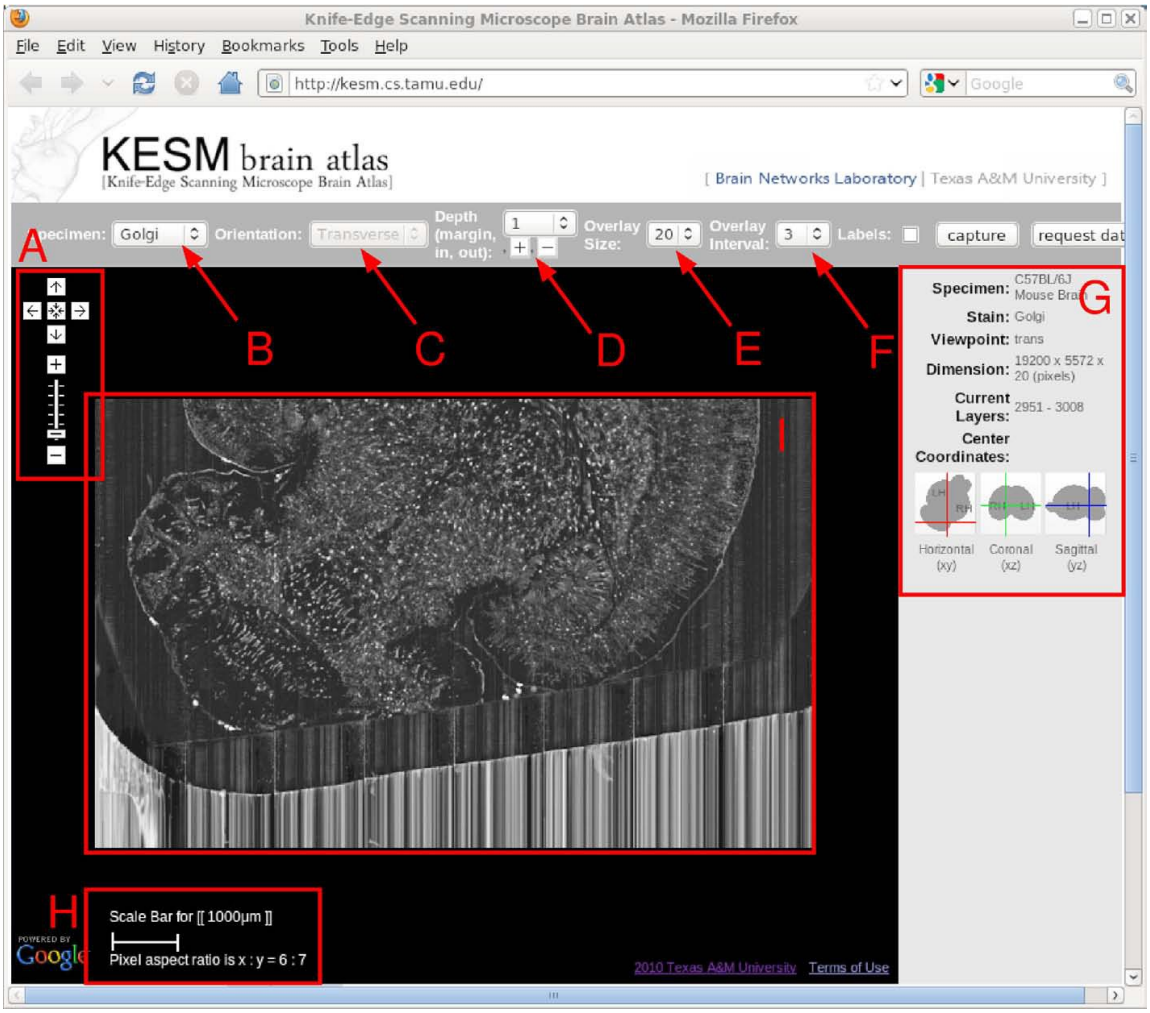


Figure 4.7: The interface of KESMBA which provides a web-based atlas service: (A) navigation control and zoom level bar, (B) and (C) data and orientation selection menu, (D) z-axis navigation control, (E) and (F) menu to seclect the number and interval of overlay, (G) infomation panel for specimen meta data and current location information, and (H) scale bar. Adapted from [8].

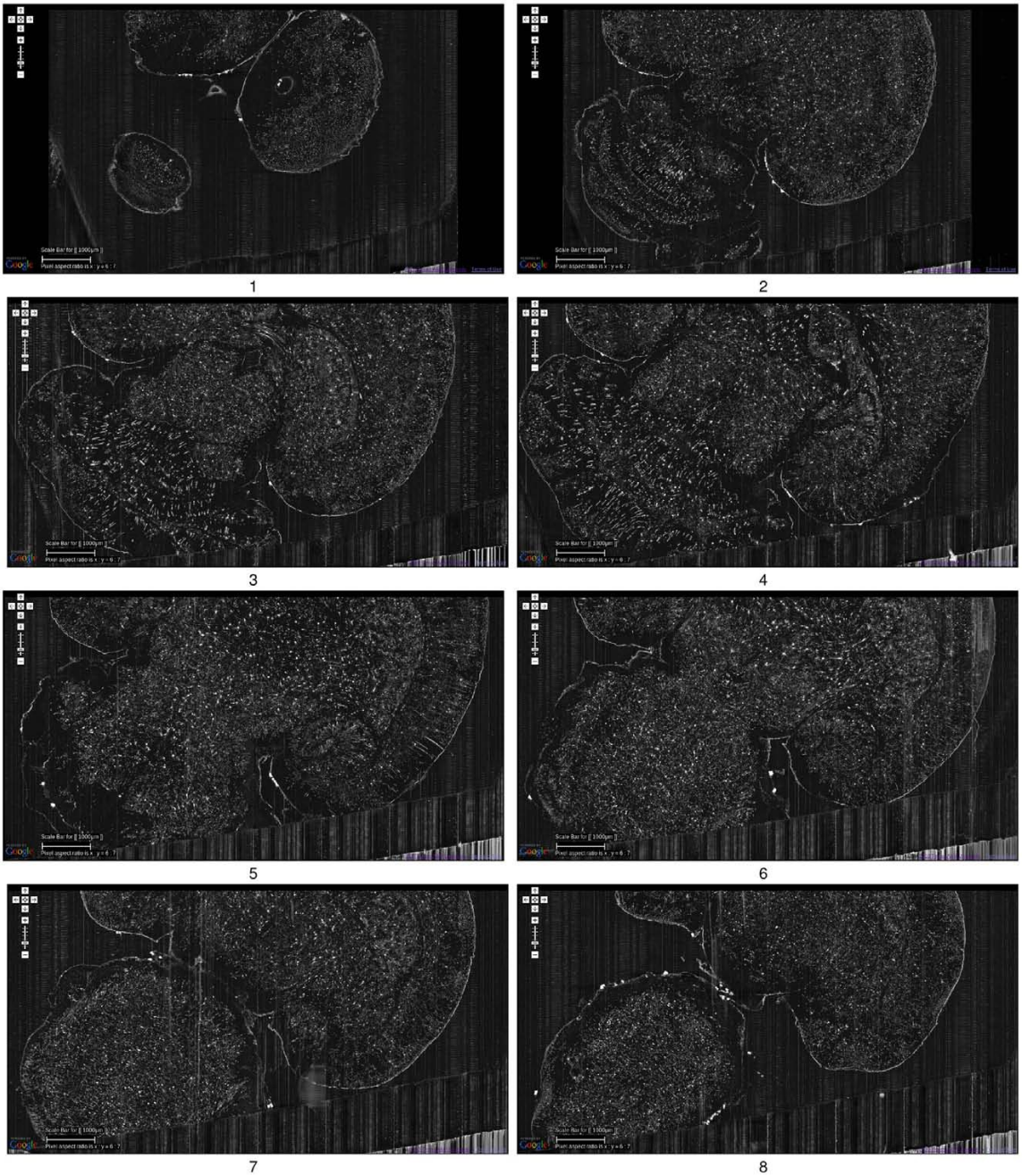


Figure 4.8: KESMBA with Golgi data set 1. The data were sectioned in the horizontal orientation (upper right corner: anterior, lower left corner: posterior). Each image is an overlay of 20 images along the z-axis. The interval between each image from (1) to (8) is $\sim 600\mu m$. These images are screenshots of the KESMBA. Adapted from [8].

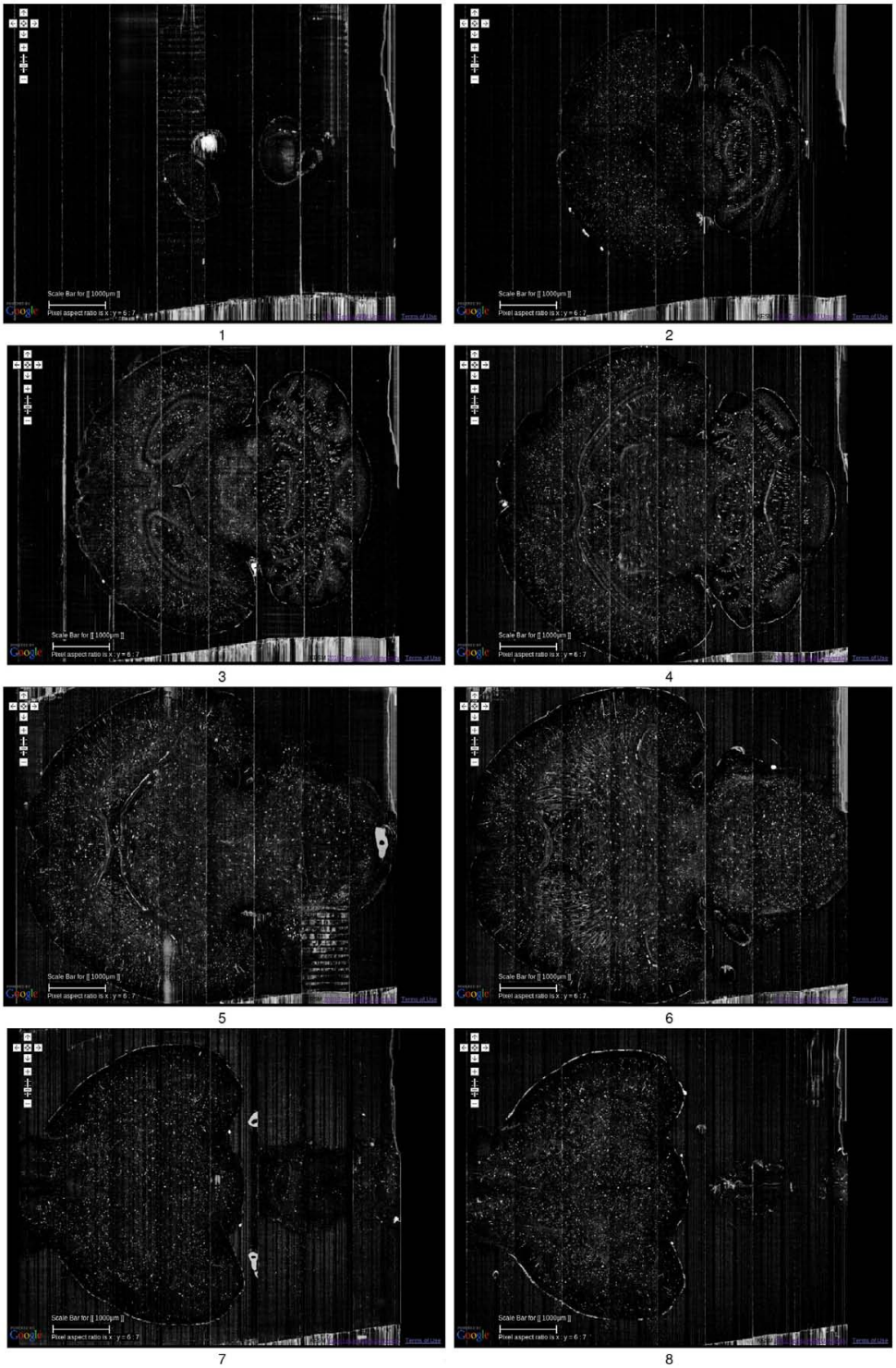


Figure 4.9: KESMBA with Golgi data set 2. The data were sectioned in the horizontal orientation (left: anterior, right: posterior). Each image is an overlay of 20 images along the z-axis. The interval between each image from (1) to (8) is $\sim 800\mu m$, except for the last where it is $\sim 200\mu m$. These are screenshots of the KESMBA. Adapted from [8].

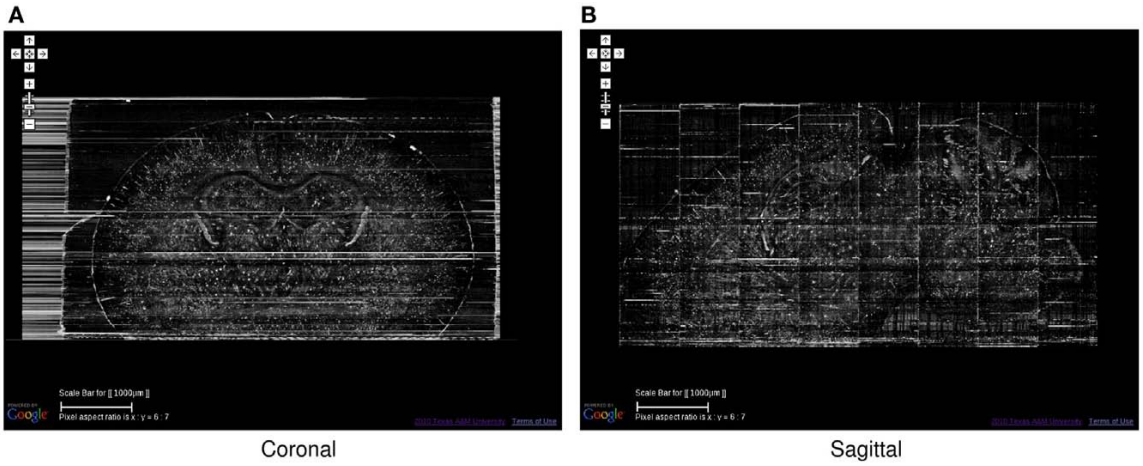


Figure 4.10: KESMBA with Golgi data set 2 in coronal and sagittal orientations: (A) coronal orientation, and (b) sagittal orientation of the same data set shown in Fig.4.9. Adapted from [8].

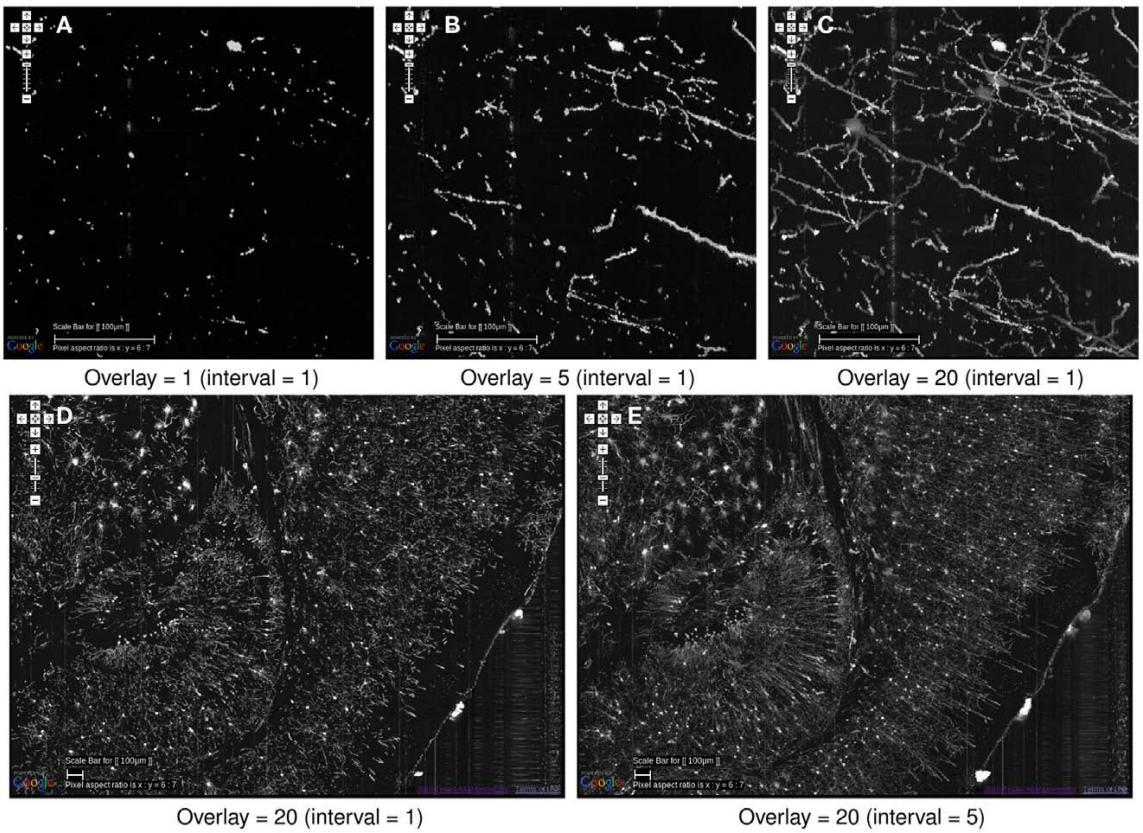


Figure 4.11: Image overlays in KESMBA Golgi data set 1. (A) A single image in $1\mu m$ thickness that provides little information about the neuronal morphology. (B)-(C) shows the effect of increasing number of overlays which corresponds to the tissue thickness. (D) and (E) shows that interval of overlaid images can be an effective method to view the neuronal circuits more clearly at a zoomed-out scale. (D) is an overlay of 20 images which corresponds to a $20\mu m$-thick tissue, and (E) is an overlay of 20 images but at an interval of 5 beteewn each layer so that it representes a $100\mu m$-thick tissue. Only in (E) we can observe 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). Adapted from [8].

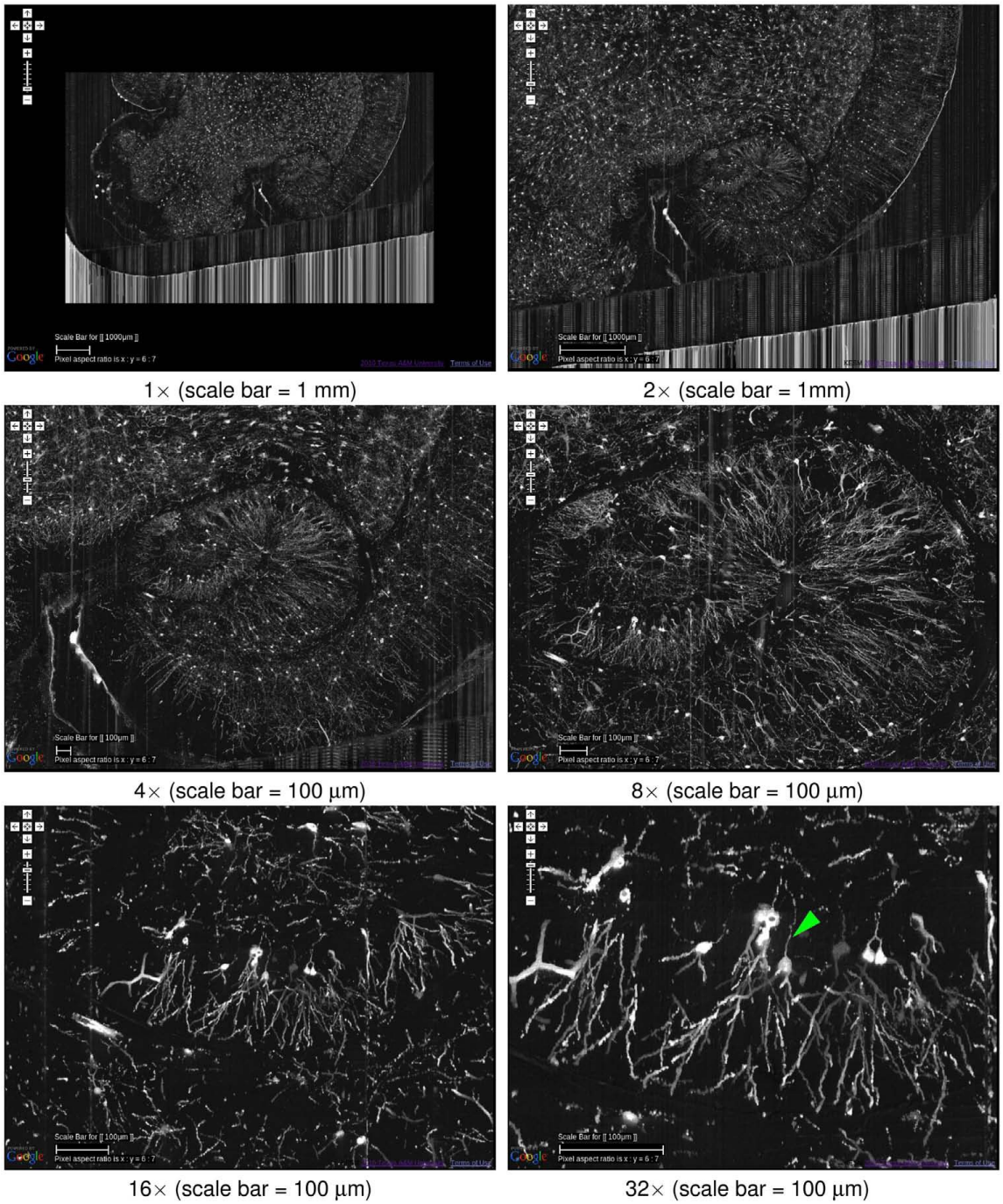


Figure 4.12: Multiscale exploration of the KESMBA. A multiscale view of the Golgi data set 1 is shown (hippocampus). The numbers below the images represent the magnification factor that correspond to zoom level 2-7, respectivley. All images are overlays of 20 images. The overay intervals were 5 in the first four images, and 1 in the last two images. The last image, 32× in zoom level 7, clearly shows axons merging from the hippocampal neurons (arrowhead). Adapted from [8].

5. KESMBA IN VECTOR GRAPHICS

The previous KESMBA is a robust tool to visualize and analyze the KESM volumetric brain image data sets, but there is still room for improvement. In the existing KESMBA, image loading is slow because of the use of the PNG image format, which is larger in file size than other raster images, due to the loss-less compression and the added alpha channel in each image (for transparency). Moreover, since Google Maps API is governed by a commercial license (Fig.5.1), it does not provide enough flexibility for customization, extension, mirroring.

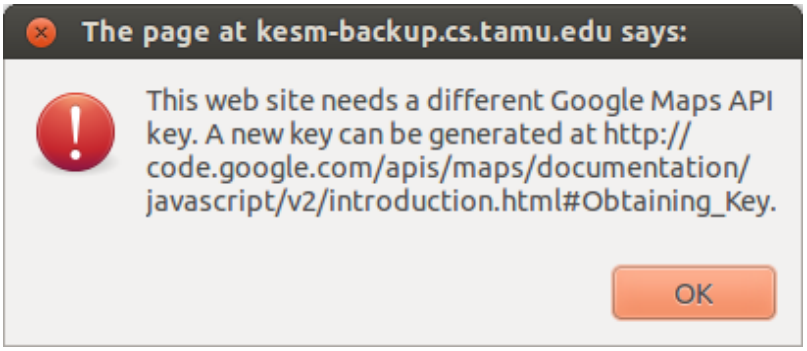


Figure 5.1: Restricted access due to the Google Maps license key. The use of Google Maps API requires a valid license key for each domain.

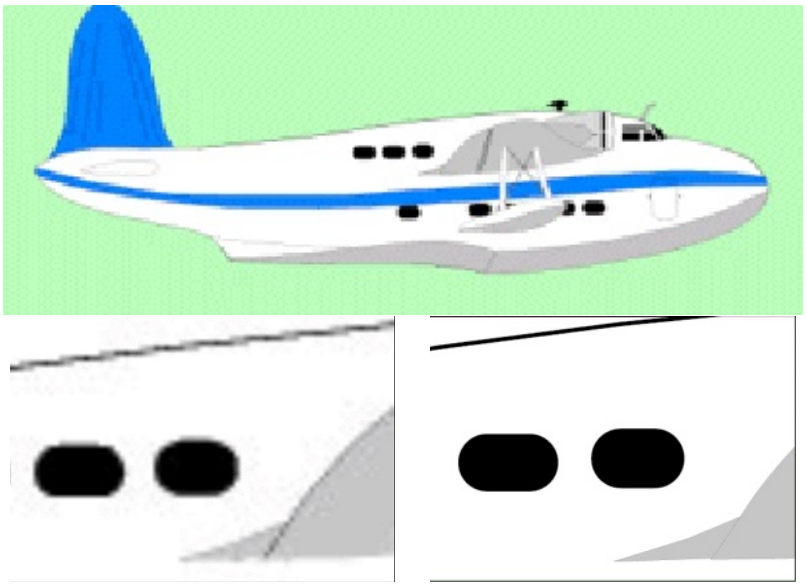
5.1 Desired properties

We need a faster and flexible framework to extend the dissemination of the KESM data. To solve these issues, we have two desired properties. The first is the reduction of the file size of each tile to increase the KESMBA access speed. We tried to find an alternative graphics format, which has smaller file size than PNG while retraining the morphological information and transparency support. The second desired property is an open source mapping framework for flexibility that is not restricted by commercial licenses and allows for open-ended customization. Consequently, we

designed and developed a new KESM mouse brain atlas that uses a vector graphics format called Scalable Vector Graphics (SVG) [32] instead of PNG, and OpenLayers API [29] instead of Google Maps API.

5.2 Vector graphics: Scalable Vector Graphics (SVG)

There are several image formats supported by web browsers, such as GIF, JPEG, PNG and so on. Most of them are pixel-based raster graphics image types and their size depend on their resolution in general. When we magnify the raster image, however, details can be quickly lost, depending on the resolution (Fig.5.2(a)). Even though still rare, web browsers are starting to support vector graphics image types. The vector format is relatively independent from pixel count and is light weight compared to the raster format, thus it can overcome the issue of increasing file size according to resolution and limitations in magnification (Fig.5.2(b)) [9].



(a) Magnification of a raster image

(b) Magnification of a vector image

Figure 5.2: Raster image versus Vector image. Adapted from [9]

A widely used vector graphics image file format is the Scalable Vector Graphics (SVG) format introduced in 1999 by the World Wide Web Consortium (W3C) [9], [30], [32]. It is a type of Extensible Markup Language (XML) to describe 2D graphics and can be displayed on most Internet web browsers. SVG defines an object as a set of lines and curves, and each image object is represented as a SVG path element. For vectorization, which is the conversion of raster graphics to vector graphics, first, the objects in the raster images need to be detected by edge-detection. Then, based on the edge-detection results, the SVG images are vectorized by representing the edge using tracing commands such as line and curve. Below is an example code of the SVG image file format, which is an XML type.

```
<?xml version="1.0" encoding="UTF-8" standalone="no"? >
<!DOCTYPE svg PUBLIC "-//W3C//DTD SVG 1.0//EN"
"http://www.w3.org/TR/2001/REC-SVG-20010904/DTD/svg10.dtd">
<svg version="1.0" xmlns="http://www.w3.org/2000/svg"
width="256" height="256" viewBox="0 0 256 256">
<path style="fill:#1e1e1e; stroke:none;"
d="M475 17.6096C468.633 19.2413 462.736 24.3162 461.41 31
C459.842 38.9017 467.161 46.2966 475 44.4421L482.057 42.7727L489.445 36.4996
C491.552 20.5464 482.103 15.7892 475 17.6096 ... z"/> </svg>
```

The SVG file code consists of a header and SVG path information. As mentioned above, the SVG paths represent the edge of the objects in the image described by tracing commands such as line and curve. Basically, each SVG path starts with a M(move to) X Y command, which means starting a new path at the given (x,y) coordinate, and the coordinate (x, y) would become the current point of the next command. L(line to) X Y means drawing a line from the current point to the given

(x,y) coordinate and C(curve to) X1 Y1 X2 Y2 X Y means drawing a curve from the current point to (x,y) using (x1,y1) as the control point at the beginning of the curve and (x2,y2) as the control point at the end of the curve. By recording the object information in the images with SVG path information, the SVG images can have a relatively small file size compared to raster images when the objects have simple shapes. If the images have many complicated and detailed objects, the SVG images can become larger in file size and can take longer to load than raster images. In our case, however, the file size of the mouse brain image data sets can be dramatically reduced when existing raster data (PNG) are converted to the vector graphics format. With the reduced data size, the SVG-based brain atlas can significantly reduce the image loading time. Due to this, the SVG graphics format is a good fit for our application domain. Moreover, the SVG format has additional advantages, such as access to individual segmented objects and the ease of labeling each of these segments.

5.3 OpenLayers API

We employed OpenLayers API [29], an open-source JavaScript library for geographical mapping to visualize the KESM data sets on the web, in place of the Google Maps API we used in the previous version of KESMBA. Since OpenLayers API is under a FreeBSD (Berkeley Software Distribution) license, it can be extended and modified by anyone. With this, we can open-endedly customize the API based on our requirements [33]. OpenLayers is already a popular open source geographic tool in the Geographic Information System (GIS) community [34], [37]. Just like Google Maps API, OpenLayers API provides enough essential functions for visualizing and navigating geographic data to implement the KESMBA such as overlaying, zooming, panning, etc. Also it is actively developed by a large community of developers.

Figure 5.3: Example map shown using OpenLayers API. Source: http://openlayers.org/.

6. IMPLEMENTATION AND RESULTS

For the second generation KESMBA, we combined SVG and OpenLayers. In this chapter, we state in detail how to covert the tiles from PNG to SVG, how to implement and customize OpenLayers for KESMBA + SVG, and show the resulting performance improvement.

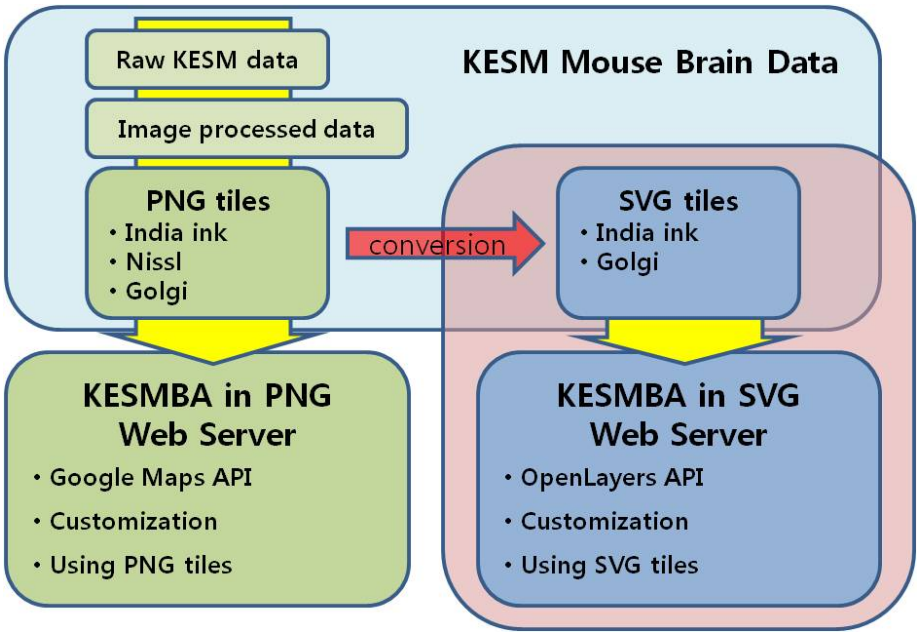


Figure 6.1: The KESM data flow and KESMBA web server

6.1 Conversion tiles

In the original KESMBA in PNG (bitmap), we already had the PNG tiles from KESM mouse brain data sets. The raw image data sets acquired from KESM were preprocessed to enhance the image quality for KESMBA: inverting image; gamma correction; and transparent background. After that, we generated the multiscale tiles for the atlas as mentioned in the Prior work (Fig.6.1).

For the SVG-based KESMBA, we converted the existing PNG tiles to SVG tiles

using Autotrace-0.31.1 [20], a bitmap to vector graphics conversion tool. Before converting, we needed to scale up the PNG tile images to retain the detailed shape information of the objects, because the scaled up images allow the interval of SVG path elements to be more dense for describing elaborate shapes. For 2 times scaling up the image, we used ImageMagick [14], which is a conversion, editing, and composing tool for bitmap images. The command is as below.

```
convert [PNG tile] -background black -adaptive-resize 200x200%!
-negate [Scaled up image]
```

After scaling up, we ran Autotrace for SVG conversion with transparent background as below:

```
autotrace -background FFFFFF -despeckle-level 1 -color-count 2 -line-threshold 0.8
-output-format svg [Scaled up image] > [SVG tile]
```

The "-background FFFFFF" option is used to make transparent background for overlaying with distance attenuation, and the "-color-count 2" option means that it performs binary segmentation.

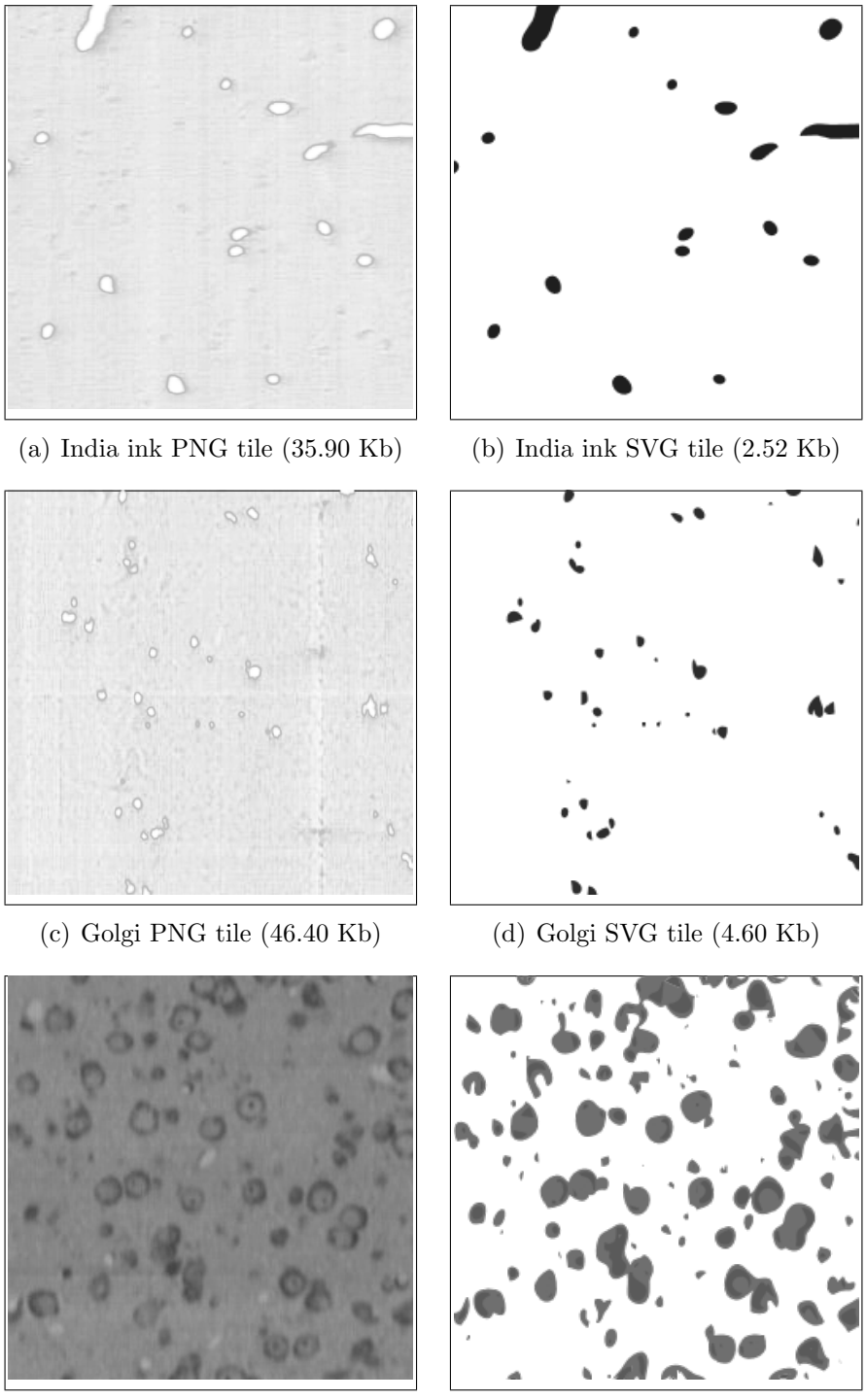
The original SVG file of the Autotrace result, however, has a non-standard SVG header.

```
<?xml version="1.0" standalone="yes"? >
<svg width="256" height="256">
```

As a result, the original SVG header leads to different rendered results on different web browsers. Therefore we had to modify the SVG header according to the current standard SVG header format. Finally, we scaled back the 2 times scaled up size to the original size by adding a tag in the SVG header. The source code of Autotrace was modified to directly generate the SVG header as shown below.

```
<?xml version="1.0" encoding="UTF-8" standalone="no"? >
<!DOCTYPE svg PUBLIC "-//W3C//DTD SVG 1.0//EN"
"http://www.w3.org/TR/2001/REC-SVG-20010904/DTD/svg10.dtd">
<svg version="1.0" xmlns="http://www.w3.org/2000/svg"
width="256" height="256" viewBox="0 0 256 256">
<g transform="scale(.50)">
```

We collected the converted SVG tiles from the entire set of PNG tiles in the existing KESM India ink and Golgi data sets (Fig.6.2 (a)-(d)). However the Nissl KESM data has enough information in each layer, thus it is not needed to overlay the images, so the PNG-based KESMBA is good enough to navigate the Nissl data without overlay. Due to this, we did not conduct image processing for transparent backgound with Nissl tiles. Moreover, the intensity variation of Nissl data have important meaning, so we cannot represent the Nissl data by simple conversion to SVG image (Fig.6.2 (e)-(f)). If we want to visualize the correct information of the Nissle data in SVG, we need more complex image enhancement, and furthermore, we have to pay the added cost of increased file size. Therefore, we did not convert the Nissl PNG tiles to SVG tiles.



(a) India ink PNG tile (35.90 Kb)

(b) India ink SVG tile (2.52 Kb)

(c) Golgi PNG tile (46.40 Kb)

(d) Golgi SVG tile (4.60 Kb)

(e) Nissl PNG tile (41.60 Kb)

(f) Nissle SVG tile (39.20 Kb)

Figure 6.2: Example of conversion from PNG to SVG

6.2 OpenLayers customization

We developed the SVG-based KESMBA using OpenLayers API. The OpenLayers API and our extensinos are shown in Fig.6.3.

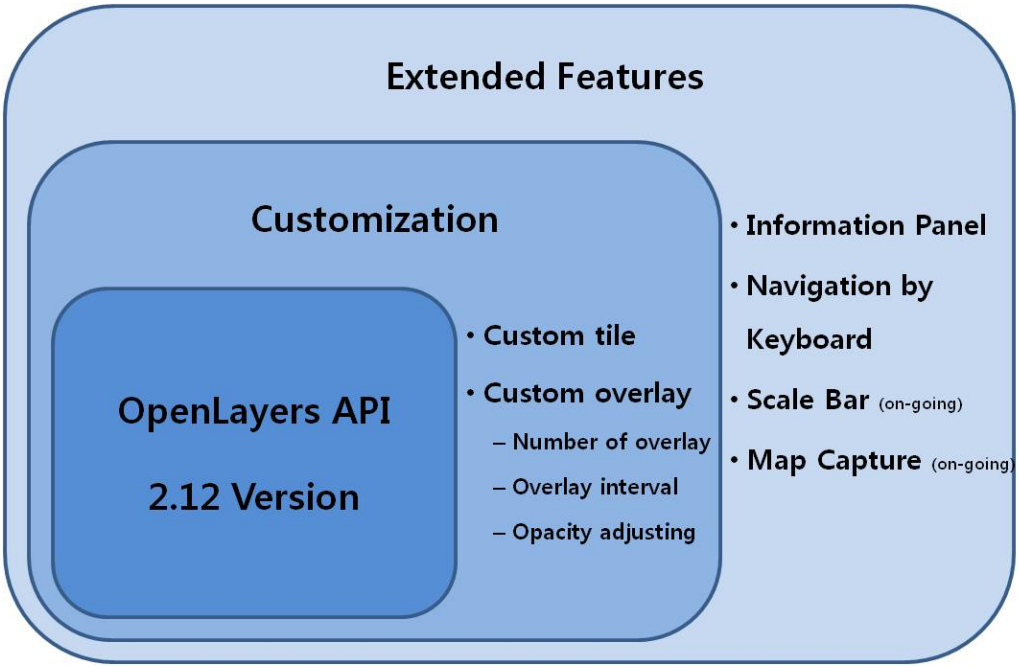


Figure 6.3: Structure of the SVG-based KESMBA using OpenLayers API.

The most important feature of KESMBA is the use of custom tiles. We customized the function of creating tile layers for use with the KESM custom tiles on the blank canvas without any geographic map. Subsequently we customized the function of overlaying layers for overlaying our custom tiles with options to select the number and interval of overlay. For distance attenuation, the overlaying function gives different opacity value to each layer to make the opacity to increase from the bottom to the top of the overlaid layer. On top of that, we added extended features: information panel, scale bar, map capture button, and keyboard navigation controller (Fig.6.4).

For the user interface, we implemented the index page using PHP. In the main page we provided the top horizontal menu bar, the map panel which shows the brain atlas with SVG tiles of the KESM data, and Information panel on the left side. The OpenLayers API was imported into the KESMBA web-server and loaded to support essential functions of geographical map, and the custome Javascripts added to the SVG-based KESMBA. Excerpts of the main page (index.php) and the layering code (svglayers.js) are listed below.

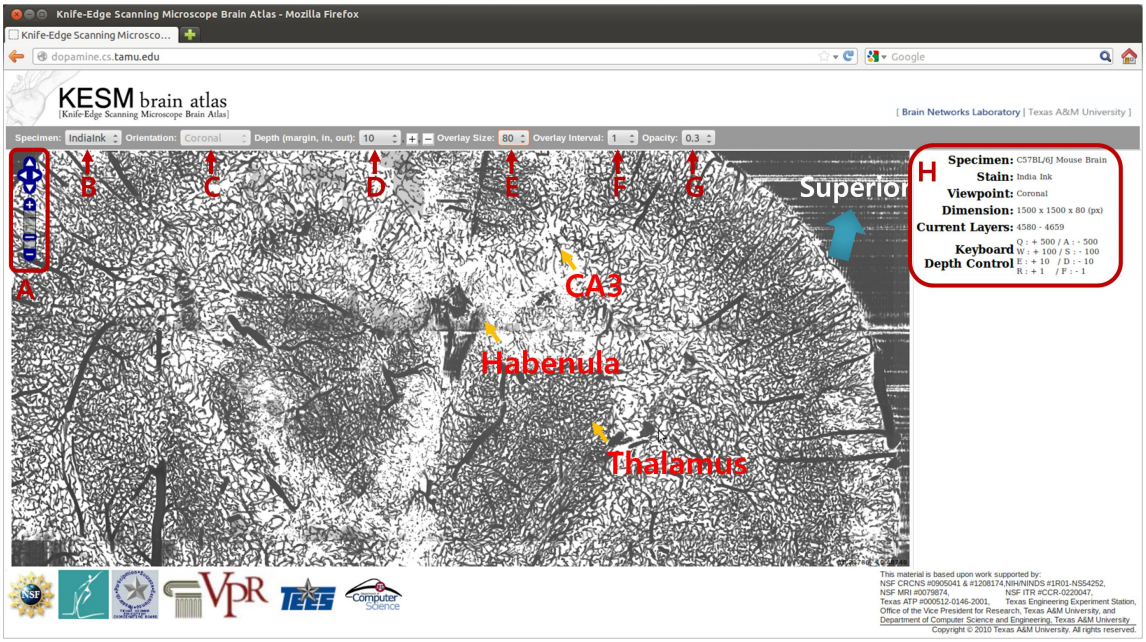


Figure 6.4: The features of the SVG-based KESMBA using OpenLayers API. (A) Navigation control and zoom level bar. (B) and (C) Data and orientation selection menu. (D) Z-axis navigation control. (E) and (F) Menu to select the number and interval of overlay. (G) Opacity option to adjust transparency level. (H) Infomation panel for specimen meta data and current location information.

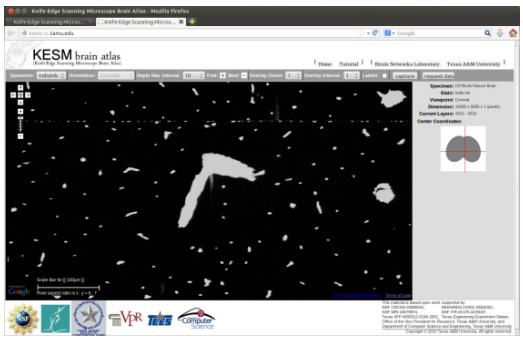
```
//////////////////////////// File: index.php ////////////////////////////
<script src="OpenLayers-2.12/OpenLayers.js" type="text/javascript"></script>
<script src="scripts/svglayers.js" type="text/javascript"></script>

// Menu bar
<div class="hor_menu">
<table class="upperMenu">
...
// 1. Data selection
<th>Specimen:</th>
<select id="specimen" name="specimen" onchange="newSpecimen(this);">
...
// 2. Orientation selection
<th>Orientation:</th>
<select id="plane" name="plane" disabled="disabled" onchange="newPlane(this);">
...
// 3. Z-axis Navigation
<th>Depth Nav. Interval:</th>
<select id="depthMargin" name="depthMargin" onchange="newDepthMargin(this);">
<img class="btnImg" id="mvIn" name="mvIn" src="./images/plus1.png"
alt="Move-in(deeper)" onmousedown="pressPlus();" onmouseup="releasePlus();"
onclick="moveIn();">
<img class="btnImg" id="mvOut" name="mvOut" src="./images/minus1.png"
alt="Move-out(shallower)" onmousedown="pressMinus();" onmouseup="releaseMinus();"
onclick="moveOut();">
...
```

```
// 4. Number of overlay
<th>Overlay Count:</th>
<select id="numOverlay" name="numOverlay" onchange="newNumOverlay(this);">
...
// 5. Overlay interval
<th>Overlay Interval:</th>
<select id="intOverlay" name="intOverlay" onchange="newIntOverlay(this);">
...
// 6. Adjust opacity for distance attenuation
<th>Opacity:</th>
<select id="valOpacity" name="valOpacity" onchange="newValOpacity(this);">
```

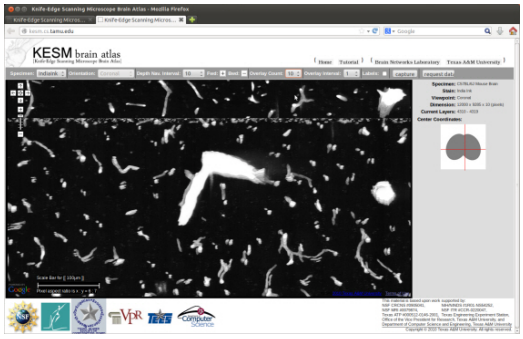
```
//////////////////////////// File:svglayers.js ////////////////////////////
// 1. Generate tile's paths function get_kesm_url (bounds) {
var x_vary = 2*Math.pow(2, z);
var y_vary = 2*Math.pow(2, z);
var cur_x = Math.round((bounds.left-this.maxExtent.left)/(this.map.getResolution()
this.tileSize.w))+(x_vary);
var cur_y = Math.round((this.maxExtent.top-bounds.top)/(this.map.getResolution()
this.tileSize.h))+(y_vary);
var cur_z = this.map.getNumZoomLevels() - (z+1);
var path = cur_z +"_" + cur_x + "_" + cur_y + ".svg";
if (url instanceof Array) url = this.selectUrl(path, url);
return url + path;}
```

```
...
// 2. Initilalize map
function svg_init(){
g_map = new OpenLayers.Map( 'svgmap', {
...
var b_layer = new OpenLayers.Layer.TMS("Name", [new_url],
{ type:'svg', getURL:get_kesm_url, opacity: g_valOpacity});
g_map.addLayer(b_layer);
...
// 3. Ovelay layers
for(var idx = 0; idx ¡ g_numOverlay-1; idx++) {
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]);
}
...
// 4. Z-axis keyboard control
function key_check(){
if(event.keyCode == ...}
...
```



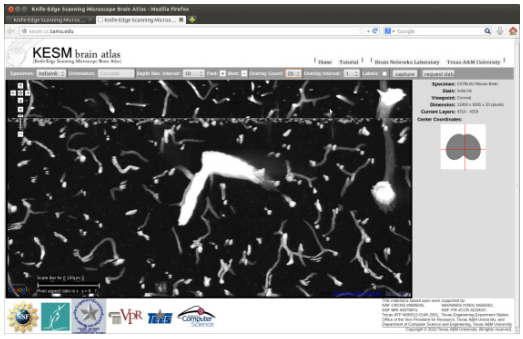
(a) A screenshot of PNG-based KESMBA with a single layer

(b) A screenshot of SVG-based KESMBA with a single layer

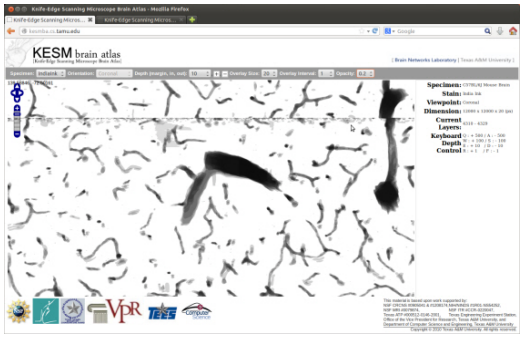


(c) A screenshot of PNG-based KESMBA with Number of ovelay = 10

(d) A screenshot of SVG-based KESMBA with Number of ovelay = 10



(e) A screenshot of PNG-based KESMBA with Number of ovelay = 20

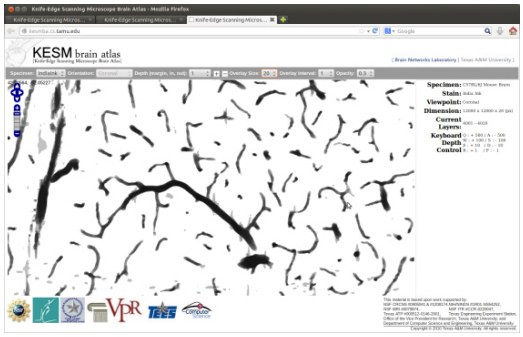


(f) A screenshot of SVG-based KESMBA with Number of ovelay = 20

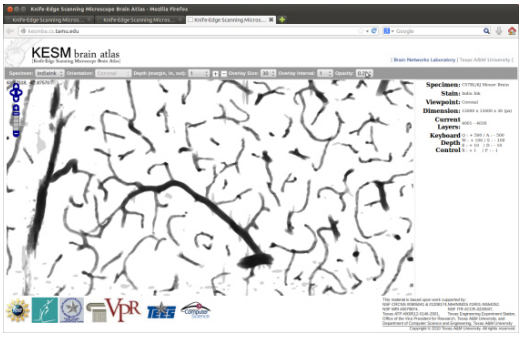
Figure 6.5: Comparison of the PNG-based KESMBA and the SVG-based KESMBA.

(a) A screenshot of SVG-based KESMBA with Number of ovelay = 5

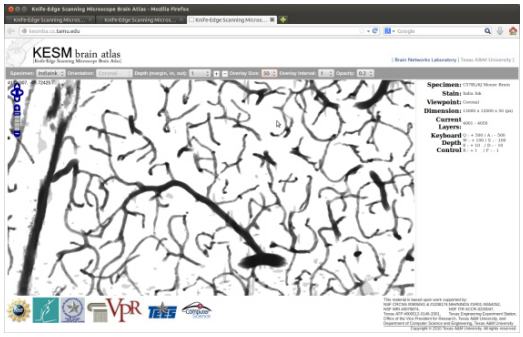
(b) A screenshot of SVG-based KESMBA with Number of ovelay = 10



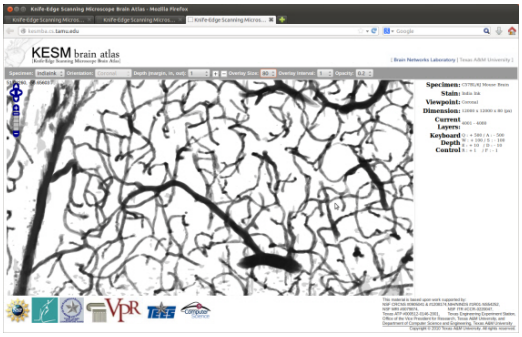
(c) A screenshot of SVG-based KESMBA with Number of ovelay = 20



(d) A screenshot of SVG-based KESMBA with Number of ovelay = 30



(e) A screenshot of SVG-based KESMBA with Number of ovelay = 50

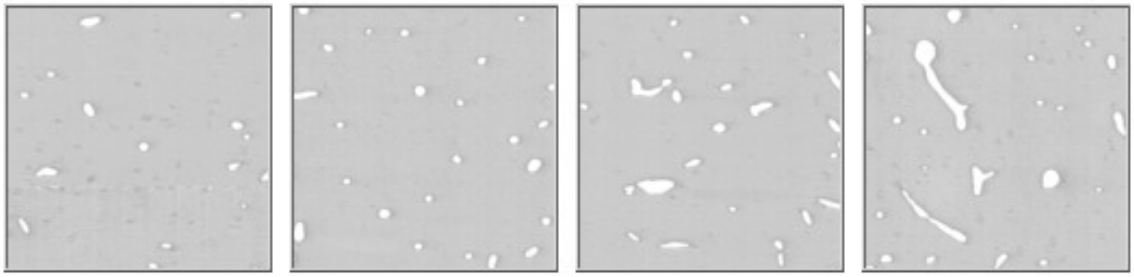


(f) A screenshot of SVG-based KESMBA with Number of ovelay = 80

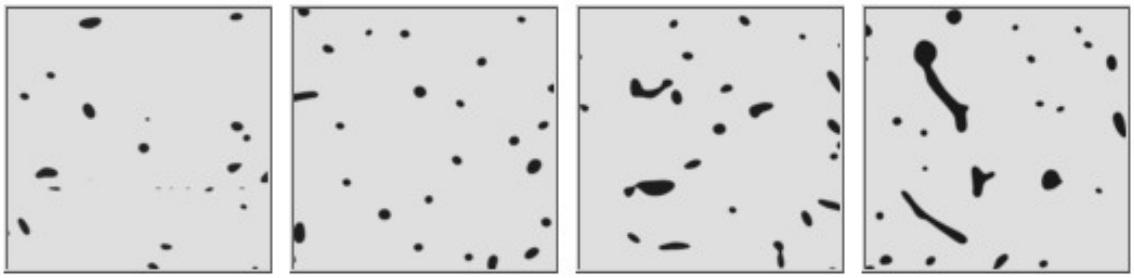
Figure 6.6: Overlay effect of the SVG-based KESMBA. The PNG-based KESMBA could overlay the KESM image data at most 40 layers, but the SVG-based KESMBA can overlay at least 2 times more layers.

6.3 Comparison of the file size of PNG VS. SVG

In this section we compare the file size of PNG and SVG tiles under various conditions. First of all, we compare the file size according to total area occupied by foreground objects and the object count. The object area is the total area of foreground objects occupying the entire image canvas and the object count is the number of objects in the image. Fig.6.7 and Table 6.1 shows the results when the total object area was varied while the object count was kept constant. Fig.6.8 and Table 6.2 shows the opposite: total object area was fixed while the object count varied. Fig.6.9 and 6.10 summarizes the results. The results show that the file size of PNG is slightly affected by both area and count, but the file size of SVG is affected more by the object count.

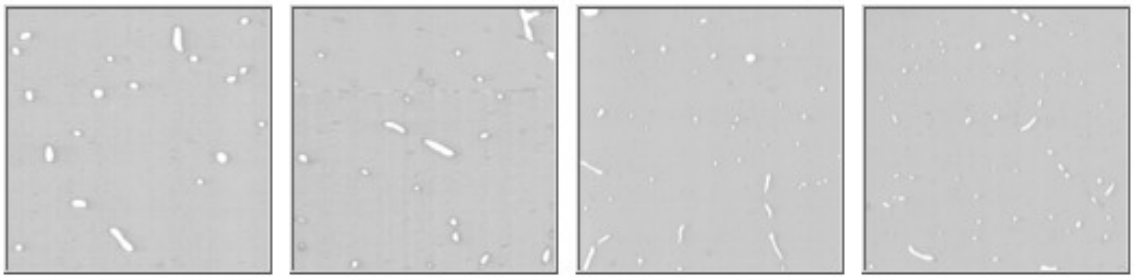


(a) PNG tiles (from left, object area: 0.019; 0.028; 0.041; 0.056. all with the same object count of 25)

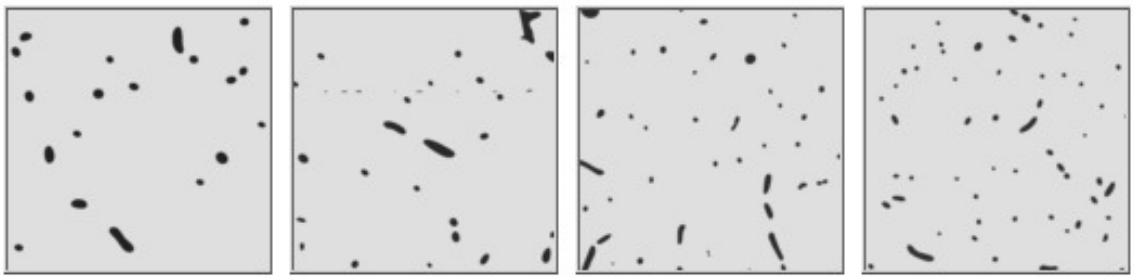


(b) SVG tiles (from left, object area: 0.019; 0.028; 0.041; 0.056. all with the same object count of 25)

Figure 6.7: Comparison of tiles with different total object area and the same object count.



(a) PNG tiles (from left, object count : 19; 33; 44; 57. with the same object area of 0.024)



(b) SVG tiles (from left, object count : 19; 33; 44; 57. with the same object area of 0.024)

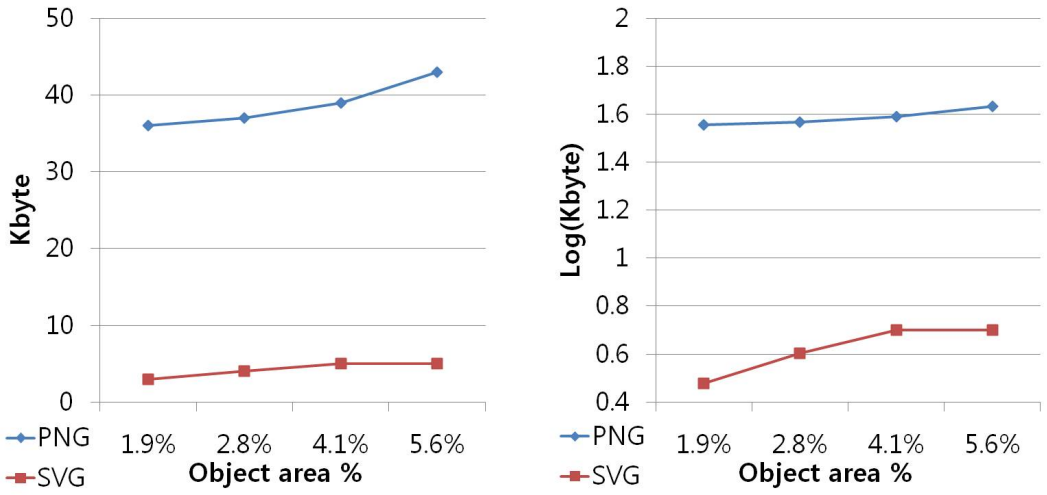
Figure 6.8: Comparison of tiles with different object count and the same total object area.

Object area	0.019	0.028	0.041	0.056
PNG	36 KB	37 KB	39 KB	43KB
SVG	3 KB	4 KB	5 KB	5 KB

Table 6.1: Comparison of file size of tiles with different total object area and the same object count.

Object count	19	33	44	57
PNG	37 KB	38 KB	45 KB	53KB
SVG	3 KB	5 KB	6 KB	7 KB

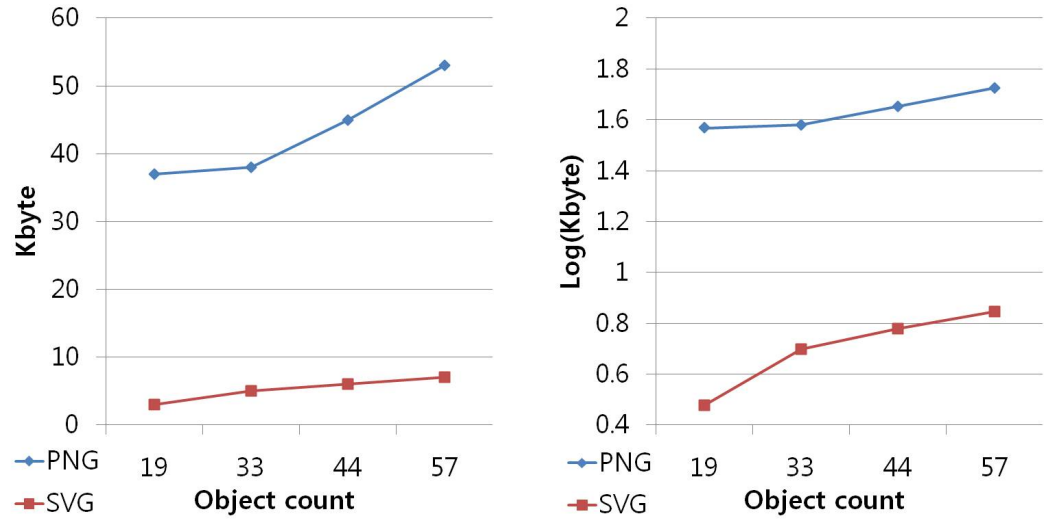
Table 6.2: Comparison of file size of tiles with different object count and the same total object area.



(a) Kbyte of Y-axis

(b) Log(Kbyte) of Y-axis

Figure 6.9: Comparison of file size of tiles with different total object area and the same object count.

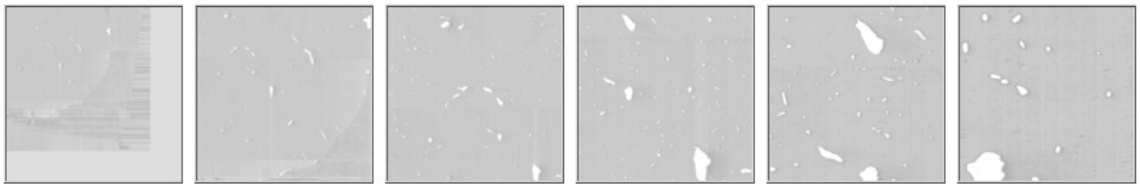


(a) Kbyte of Y-axis

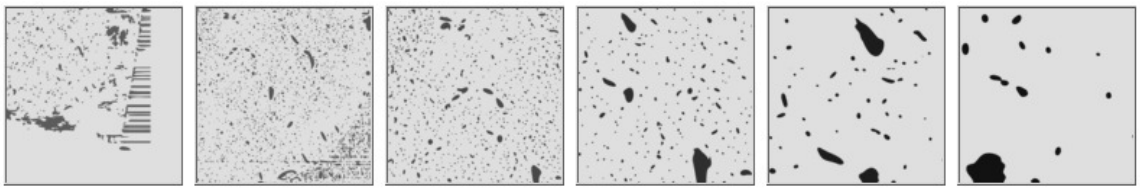
(b) Log(Kbyte) of Y-axis

Figure 6.10: Comparison of file size of tiles with different object count and the same total object area.

Next, we compared (1) the file size of a tile and (2) that of an entire layer, both as a function of the zoom level. In Fig.6.11, zoom level 2 is the entire map of the brain shown in a single screen, and zoom level 7 is the original KESM image resolution. The result of the file size of a tile in each zoom level looks nonlinear (Fig.6.12), which is affected by the number of tiles in each zoom level and margins of border tiles. However, the result of total file size per layer according to zoom level shows that the total file size per layer is increased according to increase zoom level (Fig.6.13). While the PNG version shows a dramatical increase in the total file size, the SVG version shows a more gradual increase. Finally, the total size per layer of PNG is 4 times greater than that of the SVG.



(a) Example of PNG tiles (from left, level 2 to level 7)



(b) Example of SVG tiles (from left, level 2 to level 7)

Figure 6.11: Example tiles from different zoom levels. Object count and total area vary.

	Level 2	Level 3	Level 4	Level 5	Level 6	Level 7	Total
# of tiles	9	16	36	144	576	2304	3085

(a) The total number of tiles in each zoom level

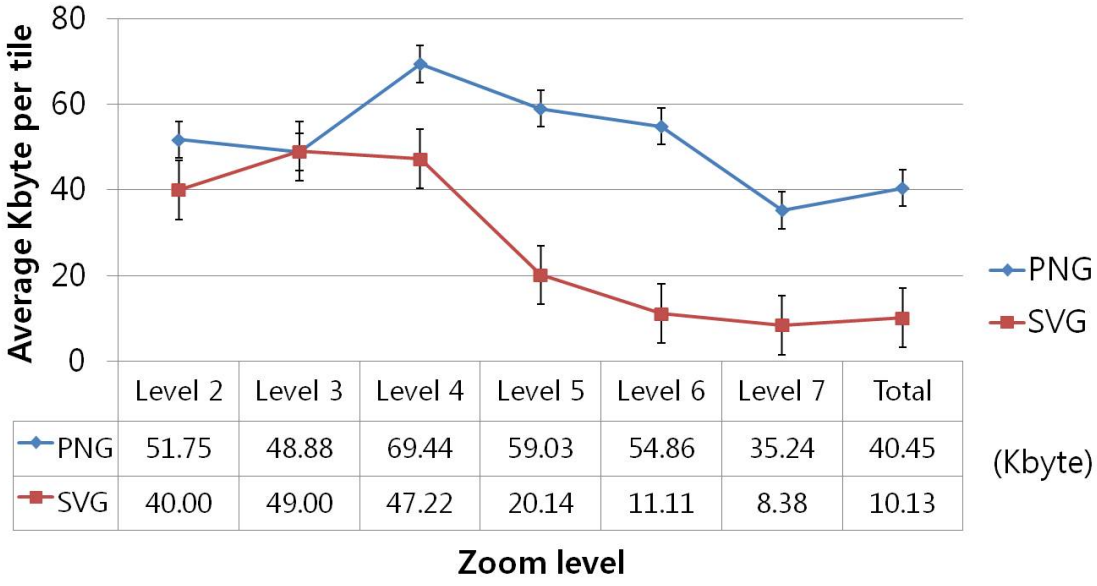
							Total avg.
PNG	51.75	48.88	69.44	59.03	54.86	35.24	40.45
SVG	40.00	49.00	47.22	20.14	11.11	8.38	10.13

(b) Average file size of a tile in each zoom level (KByte)

PNG	0.21	0.78	2.50	8.50	31.60	81.20	124.79
SVG	0.16	0.78	1.70	2.90	6.40	19.30	31.24

(c) Total file size of a layer in each zoom level (MByte)

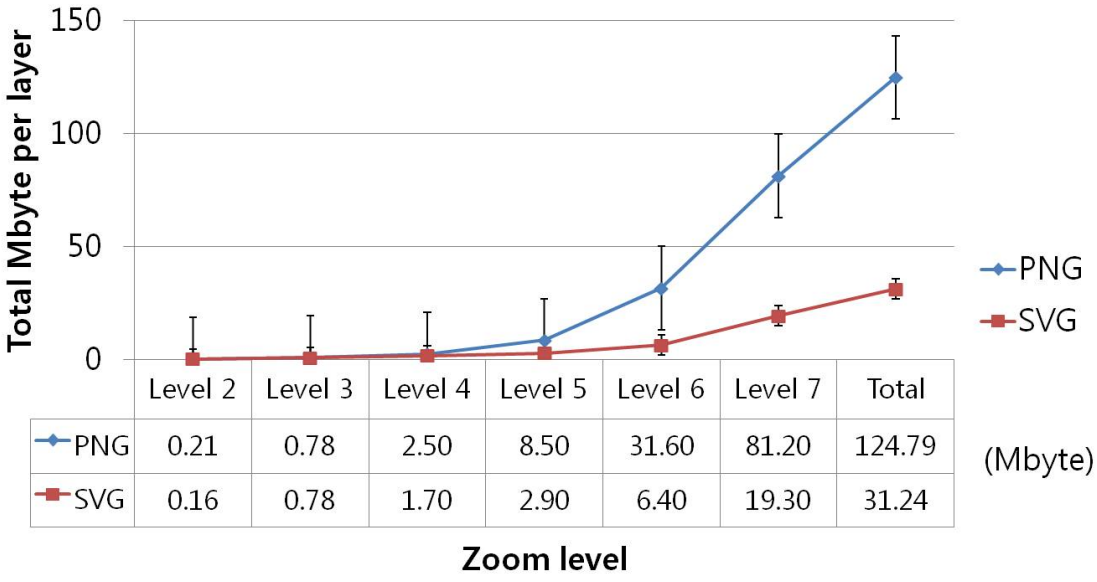
Table 6.3: Comparison of file size with respect to zoom level.



	Level 2	Level 3	Level 4	Level 5	Level 6	Level 7	Total
PNG	51.75	48.88	69.44	59.03	54.86	35.24	40.45
SVG	40.00	49.00	47.22	20.14	11.11	8.38	10.13

(Kbyte)

Figure 6.12: Average file size of a tile in each zoom level.



	Level 2	Level 3	Level 4	Level 5	Level 6	Level 7	Total
PNG	0.21	0.78	2.50	8.50	31.60	81.20	124.79
SVG	0.16	0.78	1.70	2.90	6.40	19.30	31.24

(Mbyte)

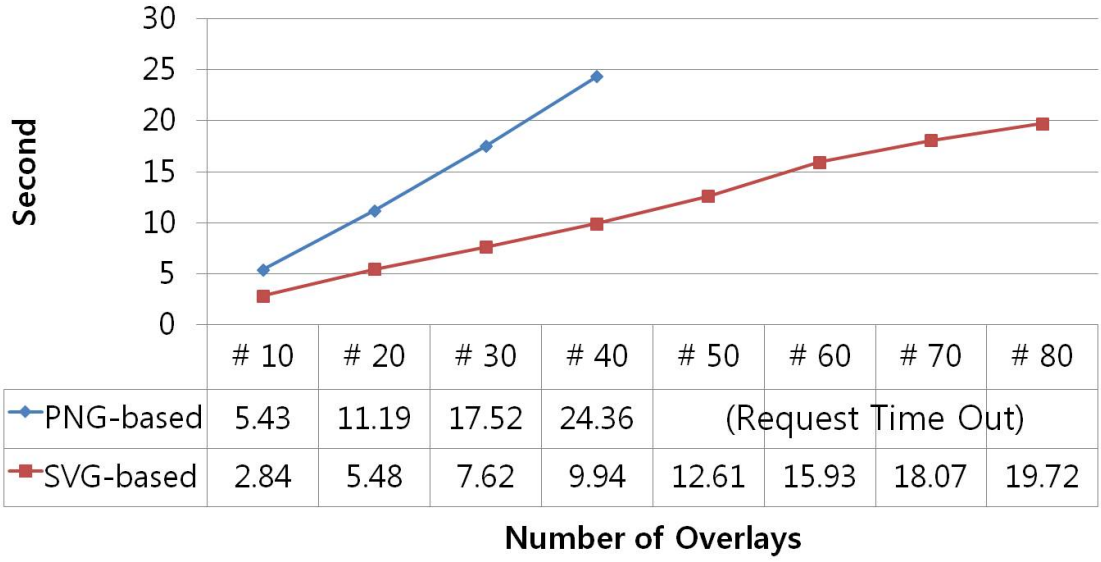
Figure 6.13: Total file size of a layer with respect to zoom level.

6.4 Improved speed of SVG-based KESMBA

To evaluate the enhanced performance, we measured the access speed of both KESMBA version. We used a tool called HttpWatch 8.5.51 (http://www.httpwatch.com/), which can measure the web page loading time in two popular web-browsers (Internet Explorer and Mozilla Firefox). The measurement condition was as follows.

- Machine: Laptop (Intel Core2 Duo 2.1Ghz, 4GB RAM, NVIDIA G102M GPU)
- Software: Windows 7 64bits, Mozilla Firefox 21.0
- Network: Sever and Client are in the same Local Network (10/100M-based)

We measured the average page loading time and image tiles 5 times per each number of overlay condition (5, 10, 20, 30, 40, 50, 80). The PNG-based KESMBA was limited to 40 overlays due to request time out when the number of overlays was more than 40.



	# 10	# 20	# 30	# 40	# 50	# 60	# 70	# 80
PNG-based	5.43	11.19	17.52	24.36	(Request Time Out)			
SVG-based	2.84	5.48	7.62	9.94	12.61	15.93	18.07	19.72

Figure 6.14: Comparison of loading time with respect to the number of overlays.

6.5 Summary

The SVG tiles of KESM data are significantly reduced (at least 4 times) in size compared to PNG tiles. Due to the reduced file size, the SVG-based KESMBA using OpenLayers is 2.45 times faster than the previous version (for number of overlays = 40). Although the download time measure of PNG-based KESMBA was limited to 40 layers we can see at the least that the loading time of SVG-based KESMBA with 80 layers is less than the loading time of PNG-based KESMBA with 40 layers, which is a great performance boost (Fig.6.14).

7. DISCUSSION

By completing the SVG-based KESMBA using OpenLayers, we have an enhanced KESMBA, which is a fast and robust tool to share and analyze the KESM mouse brain data. It will be more helpful to users including neuroscientists. In this chapter, the contributions of the SVG-based KESMBA, its limitations, and future work are discussed.

7.1 Contributions

To enhance the performance of KESMBA, I designed and implemented the SVG-based KESMBA using OpenLayers API. The major contributions of my research are: (1) Reduction the KESM data in file size by converting from PNG tiles to SVG tiles: It makes the KESMBA faster so that users can save time when navigating and exploring the KESM data. Moreover, it allows the overlay of far more than 40 layers that was the previous maximum number of overlay layers with PNG tiles. (2) Implementation of the KESMBA with OpenLayers API: Under the FreeBSD license, we can open-endedly customize the KESMBA such as adding more features based on our needs without any concern about the license key, etc.. It also allows us to distribute more easily the KESMBA framework with the reduced data size to collaborators or other sites where Internet access is slow or limited.

7.2 Limitations

The SVG-based KESMBA has some limitations we need to address: (1) The KESMBA provides 3D information of the KESM volumetric data by using distance

attenuation, not full 3D rendering. Even though the KESMBA provides views from 3 orientations and allows more than 40 the layers with SVG tiles, at times, views from every viewpoints from any direction may be desired. (2) The potentials of SVG are not fully exploited in the KESMBA: One advantage of vector graphics images is that they enable relative zoom in and zoom out, regardless of the resolution of the original image size. However, KESMBA employed the multiscale zoom level by the pyramidal tiles which may not be necessary with SVG. Another limitation is that objects that straddle across tile borders are divided into different SVG paths into the bordering tiles.

7.3 Future work

As future work, first of all, we are working on a WebGL-based 3D volume viewer for visualizing unit volumes from the KESM data. It will be compatible with the KESMBA, i.e., once we find a region of interest, we can change to 3D mode and explore that region in full 3D using the 3D volume viewer.

Second, we can fully exploit the advantages of the SVG format. As I mentioned, SVG is basically an XML file, where the objects consist of a sets of path elements, and we can control the objects style such as color or labeling by Cascading Style Sheets (CSS). It means that we can color and label specific objects. Furthermore, the objects can be connected between the series of image in the stack, both 2D vector paths and 3D vector paths, with immediate applications in 3D vector graphic rendering and neuronal / vascular tracing and cell counting.

8. CONCLUSION

The SVG-based KESMBA using OpenLayers that I developed as part of this thesis allows faster navigation and deeper exploration of KESM data sets. The SVG tiles of KESM data are significantly reduced in size compared to PNG tiles (~ 4 times reduction). Due to reduced file size, the SVG-based KESMBA using OpenLayers is 2.45 times faster than the previous version. By enhancing the performance, the users can more easily access the KESM data, leading to broader dissemination. We expect the new atlas framework to accelerate neuroscience research.

REFERENCES

[1] LC Abbott. High-throughput imaging of whole small animal brains with the knife-edge scanning microscope. In *Neuroscience Meeting Planner*, 2008.

[2] KESM Brain Atlas: Knife-Edge Scanning Microscope Brain Atlas. http://www.kesm.org/, 2013.

[3] Richard A Baldock, Jonathan BL Bard, Albert Burger, Nicolas Burton, Jeff Christiansen, Guanjie Feng, Bill Hill, Derek Houghton, Matthew Kaufman, Jianguo Rao, et al. Emap and emage. *Neuroinformatics*, 1(4):309–325, 2003.

[4] Y Choe, LC Abbott, D Han, PS Huang, J Keyser, J Kwon, D Mayerich, Z Melek, and BH McCormick. Knife-edge scanning microscopy: high-throughput imaging and analysis of massive volumes of biological microstructures. *High-Throughput Image Reconstruction and Analysis: Intelligent Microscopy Applications*, pages 11–37, 2008.

[5] Y Choe, LC Abbott, DE Miller, D Han, HF Yang, JR Chung, C Sung, D Mayerich, J Kwon, K Micheva, et al. Multiscale imaging, analysis, and integration of mouse brain networks. In *Neuroscience Meeting Planner*, 2010.

[6] Y Choe, D Han, P Huang, J Keyser, J Kwon, D Mayerich, and L Abbott. Complete submicrometer scans of mouse brain microstructure: neurons and vasculatures. In *2009 Neuroscience Meeting Planner/Chicago, IL: Society for Neuroscience, 2009. Program*, number 389.10, 2009.

[7] Yoonsuck Choe, David Mayerich, Jaerock Kwon, Daniel E Miller, Ji Ryang Chung, Chul Sung, John Keyser, and Louise C Abbott. Knife-edge scanning

microscopy for connectomics research. In *Neural Networks (IJCNN), The 2011 International Joint Conference on*, pages 2258–2265. IEEE, 2011.

[8] Ji Ryang Chung, Chul Sung, David Mayerich, Jaerock Kwon, Daniel E Miller, Todd Huffman, John Keyser, Louise C Abbott, and Yoonsuck Choe. Multi-scale exploration of mouse brain microstructures using the knife-edge scanning microscope brain atlas. *Frontiers in neuroinformatics*, 5, 2011.

[9] David Duce, Ivan Herman, and Bob Hopgood. Web 2d graphics file formats. In *Computer Graphics forum*, volume 21, pages 43–64. Wiley Online Library, 2002.

[10] Daniel Chern-Yeow Eng and Yoonsuck Choe. Stereo pseudo 3d rendering for web-based display of scientific volumetric data. In *Proceedings of the Fifth Eurographics/IEEE VGTC conference on Point-Based Graphics*, pages 73–80. Eurographics Association, 2008.

[11] Google developers Google Maps API. http://developers.google.com/maps/, 2013.

[12] Torkel Hafting, Marianne Fyhn, Sturla Molden, May-Britt Moser, and Edvard I Moser. Microstructure of a spatial map in the entorhinal cortex. *Nature*, 436(7052):801–806, 2005.

[13] Patric Hagmann, Maciej Kurant, Xavier Gigandet, Patrick Thiran, Van J Wedeen, Reto Meuli, and Jean-Philippe Thiran. Mapping human whole-brain structural networks with diffusion mri. *PloS one*, 2(7):e597, 2007.

[14] Or Compose Bitmap Images ImageMagick: Convert, Edit. http://www.imagemagick.org/, 2013.

[15] Allan R Jones, Caroline C Overly, and Susan M Sunkin. The allen brain atlas: 5 years and beyond. *Nature Reviews Neuroscience*, 10(11):821–828, 2009.

[16] Jae-Rock Kwon and Yoonsuck Adviser-Choe. *Acquisition and mining of the whole mouse brain microstructure*. Texas A & M University, 2009.

[17] Jaerock Kwon, David Mayerich, Yoonsuck Choe, and Bruce H McCormick. Automated lateral sectioning for knife-edge scanning microscopy. In *Biomedical Imaging: From Nano to Macro, 2008. ISBI 2008. 5th IEEE International Symposium on*, pages 1371–1374. IEEE, 2008.

[18] Christopher Lau, Lydia Ng, Carol Thompson, Sayan Pathak, Leonard Kuan, Allan Jones, and Mike Hawrylycz. Exploration and visualization of gene expression with neuroanatomy in the adult mouse brain. *BMC bioinformatics*, 9(1):153, 2008.

[19] Ed S Lein, Michael J Hawrylycz, Nancy Ao, Mikael Ayres, Amy Bensinger, Amy Bernard, Andrew F Boe, Mark S Boguski, Kevin S Brockway, Emi J Byrnes, et al. Genome-wide atlas of gene expression in the adult mouse brain. *Nature*, 445(7124):168–176, 2006.

[20] AutoTrace: converts bitmap to vector graphics Martin Weber. http://autotrace.sourceforge.net/, 2004.

[21] David Mayerich, L Abbott, and B McCormick. Knife-edge scanning microscopy for imaging and reconstruction of three-dimensional anatomical structures of the mouse brain. *Journal of microscopy*, 231(1):134–143, 2008.

[22] David Mayerich, Jaerock Kwon, Chul Sung, Louise Abbott, John Keyser, and Yoonsuck Choe. Fast macro-scale transmission imaging of microvascular networks using kesm. *Biomedical optics express*, 2(10):2888, 2011.

[23] BH McCormick and DM Mayerich. Three-dimensional imaging using knife-edge scanning microscopy. *Microscopy and Microanalysis*, 10(S02):1466–1467, 2004.

[24] Bruce H McCormick. Brain tissue scanner enables brain microstructure surveys. *Neurocomputing*, 44:1113–1118, 2002.

[25] Bruce H McCormick. Development of the brain tissue scanner. *Brain Networks Lab Technical Report*, 2002.

[26] Bruce H McCormick. System and method for imaging an object, June 1 2004. US Patent 6,744,572.

[27] Shawn Mikula, James M Stone, and Edward G Jones. Brainmaps. org–interactive high-resolution digital brain atlases and virtual microscopy. *Brains Minds Media*, 3:bmm1426, 2008.

[28] Shawn Mikula, Issac Trotts, James M Stone, and Edward G Jones. Internet-enabled high-resolution brain mapping and virtual microscopy. *Neuroimage*, 35(1):9–15, 2007.

[29] Free Maps for the Web OpenLayers. http://openlayers.org/, 2013.

[30] Antoine Quint. Scalable vector graphics. *Multimedia, IEEE*, 10(3):99–102, 2003.

[31] Stephan Saalfeld, Albert Cardona, Volker Hartenstein, and Pavel Tomančák. Catmaid: collaborative annotation toolkit for massive amounts of image data. *Bioinformatics*, 25(15):1984–1986, 2009.

[32] XML Graphics for the Web Scalable Vector Graphics (SVG). http://www.w3c.org/graphics/svg/, 2013.

[33] C Schmidt. Openlayers: Free maps for the web. retrieved october 22, 2009, 2008.

[34] Stefan Steiniger and Andrew JS Hunter. Free and open source gis software for building a spatial data infrastructure. In *Geospatial Free and Open Source Software in the 21st Century*, pages 247–261. Springer, 2012.

[35] Susan M Sunkin, Lydia Ng, Chris Lau, Tim Dolbeare, Terri L Gilbert, Carol L Thompson, Michael Hawrylycz, and Chinh Dang. Allen brain atlas: an integrated spatio-temporal portal for exploring the central nervous system. *Nucleic acids research*, 41(D1):D996–D1008, 2013.

[36] Issac Trotts, Shawn Mikula, and Edward G Jones. Interactive visualization of multiresolution image stacks in 3d. *NeuroImage*, 35(3):1038, 2007.

[37] Ming-Hsiang Tsou and Jennifer Smith. Free and open source software for gis education. *Unpublished White Paper Prepared for the GeoTech Center (available at http://www. iapad. org/publications/ppgis/tsou_free-GIS-for-educators-whitepaper. pdf)*, 2011.

[38] Shanmugasundaram Venkataraman, Peter Stevenson, Yiya Yang, Lorna Richardson, Nicholas Burton, Thomas P Perry, Paul Smith, Richard A Baldock, Duncan R Davidson, and Jeffrey H Christiansen. Emage-edinburgh mouse atlas of gene expression: 2008 update. *Nucleic acids research*, 36(suppl 1):D860–D865, 2008.