Image Processing Pipeline for Web-Based Real-Time 3D Visualization of Teravoxel Volumes

Akanksha Ashwini and Jaerock Kwon(✉)

Kettering University, Flint, MI 48504, USA
{ashw6475,jkwon}@kettering.edu

Abstract. With high-throughput and high-resolution imaging technologies such as Knife-Edge Scanning Microscopy (KESM), it is possible to acquire teravoxel sized three-dimensional neuronal and microvascular images of the whole mouse brain with sub-micrometer resolution. It is imperative to be able to visualize and share these teravoxel volumes efficiently, to facilitate group efforts from research communities. However, due to the immense size of the data sets, sharing and managing them have always been a big challenge. This paper describes an image processing pipeline for a real-time 3D visualization framework that allows research groups to work in collaboration. The proposed work can visualize and share terabyte-sized three-dimensional images for study and analysis of mammalian brain morphology. Although the image processing pipeline used a KESM data set to show the feasibility of it, the proposed pipeline can also be used for other larger data sets. We believe that this novel framework for Web-based real-time 3D visualization can facilitate data sharing of teravoxel volumes across research communities.

Keywords: Web-based · 3D Visualization · Teravoxel volumes
Morphology · Image stacks · Meshes

1 Introduction

Modern high-throughput and high-resolution 3D bioimaging technologies such as Knife-Edge Scanning Microscopy (KESM) [1] have enabled imaging and reconstruction of the whole mouse brain architecture at sub-micrometer resolution. KESM performs simultaneous serial sectioning and imaging of the whole mouse brain and generates data sets that include: neuronal circuits (Golgi stained), some distribution (Nissl stained), and vascular networks (India ink stained). Fig. 1 shows a recent implementation of the microscope based on KESM. The data sets are multi-scaled images, ranging from sub-cellular ($< 1\,\mu m$) to the whole organ scale ($\approx 1\,cm$). The KESM scans a $1\,cm^3$ tissue block in approximately 50 h at a resolution of $0.6\,\mu m \times 0.7\,\mu m \times 1.0\,\mu m$. It then stores the scanned biological tissue data digitally in the form of stacked 2D images, the size for which is $\approx 2\,TB$. Then, through a processing pipeline, these stacked 2D images

Y. Tan et al. (Eds.): DMBD 2018, LNCS 10943, pp. 203–212, 2018.
https://doi.org/10.1007/978-3-319-93803-5_19

are converted into volumes composed of terabytes of voxels, referred as *Teravoxel Volumes*. Due to immense size and multi-scale nature of the data set, efficiently visualizing and sharing them among research communities for analytical studies have always imposed challenges on researchers.

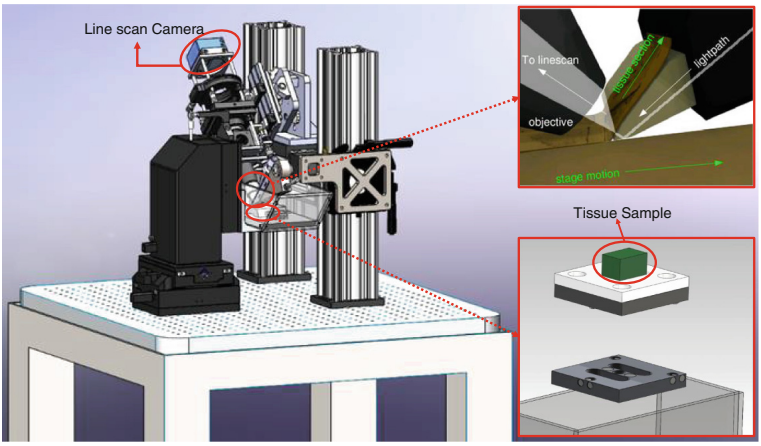

Fig. 1. The Knife-Edge Scanning Microscope. The Line-Scan Camera, Knife Assembly, Tissue Sample and Specimen Holder.

3D visualization helps users to understand the morphology of the biological organ. At the same time, efficiently sharing the data across research communities is also important for review and feedback. Terafly [2,3] is an open-source Vaa3D [4] plug-in to support 3D visualization of immensely sized biomedical images. Although the tool is efficient in visualizing terabyte-scale images, the plug-in toolkit is a stand-alone software package that is required to be installed in local computers. Also teravoxel data must reside in the same computer or local network resources. This hinders the research communities to work in collaboration with data sets. On the other hand, there have been efforts to develop web-based applications to visualize neuronal circuits and microvascular data sets (e.g. The Mouse Atlas Project, Allen Brain Atlas [5] and Knife-Edge Scanning Microscopy Brain Atlas (KESMBA) [6]). These applications allow centralization of data sets and facilitate sharing of visualization results. Yet they are not efficient real-time 3D visualization methods. For instance, Allen Brain Atlas based on the Mouse Atlas Project does not support high-resolution data visualization (details will be in the next section). KESMBA provides pseudo 3D visualization through stacking of semitransparent image slices. The maximum number of layers are 30. It takes time for KESMBA load all the layers before it displays. Also it is just layered and attenuated 2D image stacks.

In this paper, we propose an image processing pipeline for Web-based real-time framework for 3D visualization of teravoxel volumes, such as the microvascular data set obtained by the KESM. This pipeline is capable of visualizing

both terabyte volumes in real-time, since the proposed framework is Web-based, visualization is entirely independent of the underlying operating system.

2 Related Work

Mackenzie Graham developed a probabilistic atlas of the adult and developing C57BL/6J mouse. The MAP consists of not only data from Magnetic Resonance Microscopy (MRM) and histological atlases, but also a suite of tools for image processing, volume registration, volume browsing, and annotation.

The Allen Brain Atlas [7] contains detailed gene expression maps for ≈20,000 genes in the C57BL/6J mouse [5]. A semi-automated procedure was used to conduct in situ hybridization and data acquisition on 25 µm thick sections (z-axis) of the mouse brain. The x-y axis resolution of the images ranges from 0.95 µm to 8 µm.

The Mouse Brain Library (MBL) is developing methods to construct atlases from celloidin-embedded tissue to guide registration of MBL data into a standard coordinate system, by segmenting each brain in its collection into 1,200 standard anatomical structures at a resolution of 36 µm [8].

Brain Maps.org is an internet-enabled, high-resolution brain map [9]. The map contains over 10 million megapixels (35 terabytes) of scanned data, at a typical resolution of ≈ 0.46 µm/pixel (in the x-y plane). The atlas provides an intuitive Web-based interface for easy and bandwidth efficient navigation, through the use of a series of sub-sampled (zoomed out) views of the data sets, similar to the Google Maps interface.

The Whole-Brain Catalog (WBC) is a 3D virtual environment for exploring multiple sources of brain data (including mouse brain data), e.g., Cell Centered Database (CCDB), Neuroscience Information Framework (NIF), and the Allen Brain Atlas (see above). WBC has native support for registering to the Waxholm Space, a rodent standard atlas space [10].

The Knife-Edge Scanning Microscopy Brain Atlas (KESMBA) [6] framework has been designed and implemented to allow the widest dissemination of KESM mouse brain circuit data by overlaying transparent layers of images with distance attenuation. Overlaying image stacks containing two intertwining objects to get minimum intensity projection results in the loss of 3D information. Although, interleaving each image with semi-opaque blank images brings out the 3D information. Still, KESMBA provides a pseudo 3D visualization as it stacks the semitransparent image slices for not more than 30 layers at once.

Terafly is a Vaa3D plugin [3] for real-time 3D visualization of terabyte sized volumetric images. Vaa3D, which is an open-source, cross-platform system, extended its powerful 3D visualization and analysis capabilities to images of potentially unlimited size with this plugin. When used with large volumetric images up to 2.5 Terabyte in size, Vaa3D-TeraFly exhibited real-time (sub-second) performance that consistently scaled on image size. TeraFly can generate a 3D region of interest (ROI) by subsequent fetching and rendering of image data at higher resolutions, thus enabling fast (sub-second) visualization

of Terabyte-size images. It exhibits real-time performance regardless of image size when used on both high and medium-end computers. However, the performance is constrained on a local computer and cannot be directly used to share 3D visualization results of terabyte-sized data sets among research groups.

3 Background

The mouse brain data set that we used for our experiment is from a C57/BL6 mouse specimen. It is a *India Ink* stained vascular network data set, labeled with a mouse brain id: `MOU1_BRA_IND_2008_04`.

The Insight Segmentation and Registration Toolkit (ITK) is an open-source software toolkit, implemented in C++, that provides algorithms for performing registration and segmentation to multidimensional data. Segmentation is the process of identifying and classifying data found in a digitally sampled representation. Registration is the task of aligning or developing correspondences between data. ITK is widely used for medical image processing. However, It does not include methods for displaying images, nor a development environment or an end user application for exploring the implemented algorithms.

The Visualization Toolkit (VTK) is an open-source, freely available software system for scientific visualization, information visualization, 3D computer graphics, modeling, image processing, and volume rendering. VTK is implemented as a C++ toolkit, requiring users to build applications by combining various objects into an application.

The X Toolkit (XTK) is an emerging technology, including webGL and increased performance of JavaScript engines, which allows both 2D and 3D image manipulation. XTK [11] and BrainBrowser [12] are two popular tools that allow visualization and interaction with both 2D images (texture files .png, .jpg) and 3D volumes. XTK is available on GitHub and can be used to visualize a wide spectrum of physiological phenomena ranging from white matter cortical connections, aneurysm characteristics, and knee morphologies. XTK also supports a number of common neuroimaging formats such as compressed and uncompressed formats of DICOM files (.nrrd, .nii, .nii.gz, .mgz, .dcm, etc.) and files from higher level MR processing (.trk, .stl, .fsm., .label, etc.) commonly used in image analysis research. Several implementations of XTK include the AneuRisk Web repository [13] and SliceDrop.org. The LONI group (formerly of UCLA, now of UCSC) has developed an extension of SliceDrop that further supports drawing ROIs directly within the XTK framework. A pediatric brain atlas, built using the XTK visualization platform, further demonstrates the power of this framework. Another notable Web-based viewer is Papaya, based on a similarly functioned Java client.

4 Methods and Implementations

The complete implementation is broadly divided into two categories: *first:* Creation of Multidimensional Dataset that includes: (i) Processing of 2D stacked

images and (ii) Unit-Volume creation stages. *Second:* Implementation of the Web-Based 3D Visualization Framework that again comprises of two stages: (i) Web-Based GUI application and (ii) ROI selection algorithm. The first category involves a development of the image processing pipeline in Qt framework using toolkits: ITK and VTK. The second category involves the development of the web application using XTK. Let's go into details about each stage.

4.1 Processing of 2D Stacked Images

This stage involves processing of 2D images initially received from KESM, and is divided into two sub-stages:

Stitching of the Columns. After the serial sectioning performed in KESM, the raw data set of 2D images is stored in the form of stacks and columns. The tissue area from each raw image is automatically cropped and saved, so each column contains images of only tissue area from the mouse brain. The images in each column are further processed; the noise is removed, and image intensity is normalized for each cropped image. Then the images at the same z coordinate are stitched across the columns in an image sheet. ITK filters are utilized to perform the stitching algorithm. Whereas, we implemented the image intensity normalization algorithm previously proposed for KESM image stacks [14]. The outcome of this process is a single stack of 2D images with clean prominent tissue areas. For this experiment, the raw images we had were initially divided into four columns or four stacks of images. The resolution of each image in each column stack was 2400×12000. After the stitching, we finally got a stack of 9626 images each of resolution 9600×12000.

Sub-sampling of Image Stacks. To create multi-resolution image stacks, we need to sub-sample the original 2D image stack obtained after stitching. We use ITK filters to create output image stacks, which are half the resolution of the input image stack. We shrink the 2D images in the x-y plane, and choose to keep alternate files from the original stack in the z direction. So, if the resolution of the image stack created after stitching is A, the sub-sampled image stack will be of resolution A/2. This method is used to create a series of image stacks of resolution: A, A/2, A/4, A/8, A/16, A/32. For example, the original stack with 9626 images each of resolution 9600×12000 when subsampled, creates a stack with 4813 images each of resolution 4800×6000. We continuously subsample the image stacks until an image resolution lesser than 512×512 is achieved. Finally, we end up creating image stacks of resolution: 9600×12000, 4800×6000, 2400×3000, 1200×1500, 600×750, 300×375. The subsampled image stacks stored in directories are named in a way that they provide information about their resolution and the mouse brain id.

4.2 Unit-Volume Creation

This stage creates 3D volumes from the 2D image stacks created in the previous step, and is also composed of two sub stages:

Cropping of Unit-Volume Image Stacks. In this process, we crop out stacks of 256 images each of resolution 256×256 from an original image stack, which will be used later to create *unit-volumes*. 2D images of resolution 256×256, referenced as *unit-images* are cropped and extracted from each image of an input image stack. Each *unit-image* is labeled with its *unit-x* and *unit-y* coordinates. Where, *unit-x* and *unit-y* mark 256 pixels in x and y direction. Each image in an input image stack points to a coordinate in the z direction, which is used to name the directory storing all the *unit-images* for that image. Now, these directories save the *unit-images* temporarily, until they are moved to their respective *unit-volume* image stack directories (explained in further sections). Every 256 images in the input image stack marks a unit in z direction. Starting from the first image in the input image stack, a set of 256 images are taken in sequence, to create 256 temporary directories. These 256 temporary directories will create *unit-volume* image stacks of the same *unit-z* coordinate. This process is repeated for all the images in the input image stack, taken in a set of 256 files at once.

STL Mesh Creation. In this stage, we convert the *unit-volume* image stacks into volumes; create 3D STL (Standard Tessellation) meshes which can be loaded easily to the Web-based framework. We choose the volume output format as STL because it is supported by the XTK APIs. Using ITK classes, we create 3D volumetric images from the *unit-volume* image stacks. These 3D volumetric images generate *unit-volumes*, and are expected to be isotropic in all the three directions. Iso-surfaces for each *unit-volume* are found using the Marching Cube algorithm in VTK, and are then saved as a 3D mesh in an STL file format. A *STL Mesh* file is labeled with the name of its *unit-volume* image stack directory. So, for the highest resolution image stack, we create *Meshes*: `Vol_0_0_0.stl`, `Vol_0_1_0.stl`, . . . `Vol_36_45_37.stl`. This process is repeated for all the sub-sampled *unit-volume* image stacks, resulting in a set of multi-resolution *unit-volume meshes*. *Iso-Surfaces* are the distribution of scalar data in a volumetric image. Marching Cube algorithm uses patterned cubes or isosurfaces to approximate contours in a volumetric image. VTK supports the marching cubes algorithm with *VtkMarchingCubes* class, which requires a volumetric image input as a VTK data object, and creates an output in VTK poly data format. We can specify the threshold value and the number of contours while using VtkMarchingCubes, to generate the 3D surface of the object.

The Image Processing pipeline was implemented using a *controller-worker* model. Each stage explained above has its own *controller* and *worker*. A *worker* is the one which make changes; execute functions utilizing ITK and VTK libraries. Whereas, a *controller* is the one which decides what files are to be passed to the *worker* and even controls the *worker* execution. This model has been applied to avoid time delays, generally caused due to looping over the same task. Here, each time a task is to be executed, the *controller* code simply calls the *worker* application which performs that task. Since all the *controllers* are written in Qt and do not use any ITK/VTK libraries, the complete design is a cross-platform solution.

4.3 Web-Based Graphical User Interface Design

At this stage, we design the Web-based application for managing visualization and interaction with the 3D meshes. We utilized the X Toolkit (XTK): *WebGL for Scientific Visualization* [15] to make the Web-based 3D visualization framework for teravoxel volumes. A simple Web server is implemented by using *Node.js* [16]. Then through a custom javascript code, we designed the graphical user interface to control the whole functionality of the Web application. The graphical user interface (GUI) is designed using a javascript controller library called *dat.GUI* [17]. The graphical user interface is used to display the details of the volume loaded: *x,y,z* unit coordinates and the resolution.

4.4 STL Loading and ROI Selection

STL Loading. XTK supports the STL file format and for visualization, it simply requires dropping of the STL meshes on the website. X toolkit provides a set of easy-to-use APIs for visualizing scientific data on the Web. We utilized the *X.Mesh()* API to create mesh objects, for loading the *unit-volume* STL meshes created earlier. At an instance, an STL mesh of *unit-volume* i.e. of 256^3 dimension is loaded, according to the user scroll input depending upon their region-of-interest and angle-of-perspective.

ROI Selection Algorithm. This algorithm resembles the one used in the Vaa3D plugin *Terafly* or the Google Earth application. The volume of the smallest resolution i.e. $300 \times 375 \times 300$ is displayed at first. This Mesh shows the whole structure of the mouse brain architecture. Further, according to the zoom in/out levels and user scroll input, the switching between the resolution occurs. When that happens, the region-of-interest (ROI) in the next resolution is selected and displayed on the Web-server.

5 Results

The stitching operation is performed using a *Image-Stitcher Controller*, which stores the 9,626 stitched images in the directory with label indicating the mouse brain id and resolution i.e. `MOU1_BRA_IND_2008_04_9600x12000`. The *Sub-Sampling Controller* creates the following directories as shown in Table 1.

All these directories cumulatively occupy a hard disk space of ≈70 GB. The *Volume-Maker Controller* creates the following number of *unit-volume* image stacks:

All these *unit-volume* image stacks cumulatively occupy a hard disk space of ≈94 GB. Since, the iso-surfaces are only generated for volumteric images with tissue areas. Thus, the number of STL Meshes created is not always equal to the number of *unit-volume* image stacks. A *3D-Model-Maker Controller* converts the volumetric images into *STL-Meshes* and the output is displayed in Table 2 below.

Table 1. The numbers of files in sub-sample directories

Subsampled directories created	No. of files
MOU1_BRA_IND_2008_04_4800x6000	4813
MOU1_BRA_IND_2008_04_2400x3000	2407
MOU1_BRA_IND_2008_04_1200x1500	1204
MOU1_BRA_IND_2008_04_600x750	602
MOU1_BRA_IND_2008_04_300x375	301

Table 2. The numbers of images and meshes for each of resolutions

Resolution of source images	Unit-volume images	STL Meshes
9600 × 12000	57944	49884
4800 × 6000	7866	7055
2400 × 3000	990	904
1200 × 1500	80	74
600 × 750	8	8
300 × 375	1	1

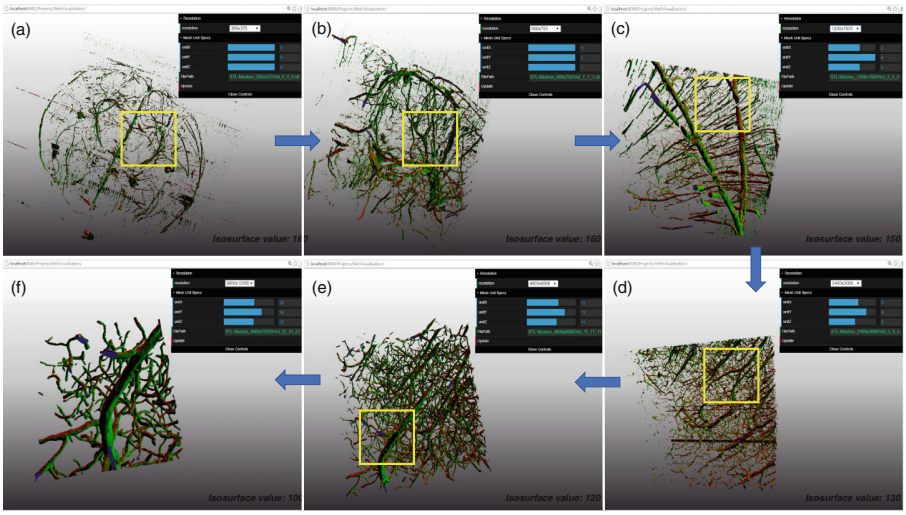

Fig. 2. The STL Meshes loaded according to the ROI selected by the user. (a) *unit-volume* loaded with *unit x-y-z* coordinates as 0,0,0 and resolution 300 × 375, (b) *unit-volume* loaded with *unit x-y-z* coordinates as 1,1,1 and resolution 600 × 750, (c) *unit-volume* loaded with *unit x-y-z* coordinates as 2,4,2 and resolution 1200 × 1500, (d) *unit-volume* loaded with *unit x-y-z* coordinates as 5,8,5 and resolution 2400 × 3000, (e) *unit-volume* loaded with *unit x-y-z* coordinates as 11,17,11 and resolution 4800 × 6000, and (f) *unit-volume* loaded with *unit x-y-z* coordinates as 22,34,22 and resolution 9600 × 12000.

All the STL's cumulatively occupy a hard disk space of ≈1 TB. A minimum of 8 GB RAM is required for creating the complete data set on a 64-bit computer architecture. It was observed that the programs written to prepare the data set work faster with such a system configuration. The framework automatically loads the smallest resolution volume when first launched. The resolution and unit x-y-z coordinates can then be selected to load the other higher resolution volume *Meshes* (See Fig. 2). It is observed that without caching, the time taken to display the data is about 10–15 s.

6 Conclusion and Future Work

To be able to explore the biological structure and understand the morphology, efficient 3D visualization for biomedical data sets is crucial. However, it is also important that we should be able to share the data set and visualization result with researchers working in imaging laboratories across the globe. This helps in working in collaboration for analysis studies and review. The above results demonstrate that this framework is potentially efficient in visualizing multi-resolution volumes and also scaling to larger images. The framework explained here exhibits real-time properties and caters both the demand for efficient visualization and sharing of visualization results. We expect the proposed Web-based real-time 3D visualization framework to enhance the accessibility of teravoxel volumes and help research communities to use the data sets.

We used KESM vascular data set for our experiment, but this approach is expected to support any data set of any size. The current GUI implementation requires a user to select the resolution and the *unit-x, unit-y and unit-z* coordinates for visualizing a particular *unit-volume* of interest. But, we are working to extend its capabilities, to automate this process of picking up the STL *meshes*, according to the user's mouse scroll input. After which, a user will be able to explore the whole-mouse-brain-structure in an easy and more natural way. The implementation would require a user to select the region-of-interest (ROI) by double-clicking/scrolling on a portion of the volume they want to visualize in detail. The double-click/scroll operation would then load the higher resolution STL *mesh* for the same ROI, according to the manipulated coordinates. Zoom in/out will help to explore the volume in one resolution but double clicking/scrolling in a particular region will load a higher resolution volume for the same ROI. The double click operation will work in both directions; for switching to higher resolutions as well as lower resolutions. The GUI panel would display the result of the visualization which includes resolution, unit coordinates, the number of branches, thickness, and other details.

References

1. Mayerich, D., Abbott, L., McCormick, B.: Knife-edge scanning microscopy for imaging and reconstruction of three-dimensional anatomical structures of the mouse brain. J. Microsc. **231**(1), 134–143 (2008)
2. Peng, H., Bria, A., Zhou, Z., Iannello, G., Long, F.: Extensible visualization and analysis for multidimensional images using Vaa3D. Nat. Protoc. **9**(1), 193–208 (2014)
3. Bria, A., Iannello, G., Peng, H.: An open-source Vaa3D plugin for real-time 3D visualization of terabyte-sized volumetric images. In: IEEE 12th International Symposium on Biomedical Imaging (ISBI) 2015, pp. 520–523. IEEE (2015)
4. Long, F., Zhou, J., Peng, H.: Visualization and analysis of 3D microscopic images. PLOS Comput. Biol. **8** (2012)
5. Lein, E.S., Hawrylycz, M.J., Ao, N., Ayres, M., Bensinger, A., Bernard, A., Boe, A.F., Boguski, M.S., Brockway, K.S., Byrnes, E.J., et al.: Genome-wide atlas of gene expression in the adult mouse brain. Nature **445**(7124), 168–176 (2007)
6. Chung, J.R., Sung, C., Mayerich, D., Kwon, J., Miller, D.E., Huffman, T., Keyser, J., Abbott, L.C., Choe, Y.: Multiscale exploration of mouse brain microstructures using the knife-edge scanning microscope brain atlas. Front. Neuroinformatics **5** (2011)
7. MacKenzie-Graham, A., Jones, E.S., Shattuck, D.W., Dinov, I.D., Bota, M., Toga, A.W.: The informatics of a c57bl/6j mouse brain atlas. Neuroinformatics **1**(4), 397–410 (2003)
8. Rosen, G.D., Williams, A.G., Capra, J.A., Connolly, M.T., Cruz, B., Lu, L., Airey, D.C., Kulkarni, K., Williams, R.W.: The mouse brain library @ www.mbl.org. In: Int Mouse Genome Conference, vol. 14, p. 166 (2000)
9. Mikula, S., Trotts, I., Stone, J.M., Jones, E.G.: Internet-enabled high-resolution brain mapping and virtual microscopy. Neuroimage **35**(1), 9–15 (2007)
10. Johnson, G.A., Badea, A., Brandenburg, J., Cofer, G., Fubara, B., Liu, S., Nissanov, J.: Waxholm space: an image-based reference for coordinating mouse brain research. Neuroimage **53**(2), 365–372 (2010)
11. Haehn, D., Rannou, N., Ahtam, B., Grant, E., Pienaar, R.: Neuroimaging in the browser using the x toolkit. Front. Neuroinformatics **101** (2014)
12. Sherif, T., Kassis, N., Rousseau, M., Adalat, R., Evans, A.C.: Brainbrowser: distributed, web-based neurological data visualization. Front. Neuroinformatics **8** (2014)
13. Gutman, D.A., Dunn Jr., W.D., Cobb, J., Stoner, R.M., Kalpathy-Cramer, J., Erickson, B.: Web based tools for visualizing imaging data and development of xnatview, a zero footprint image viewer. Front. Neuroinformatics **8**, 53 (2013)
14. Choe, Y., Mayerich, D., Kwon, J., Miller, D.E., Sung, C., Chung, J.R., Huffman, T., Keyser, J., Abbott, L.C.: Specimen preparation, imaging, and analysis protocols for knife-edge scanning microscopy. J. Visualized Exp. JoVE **58** (2011)
15. XTK the x toolkit: WebGL for scientific visualization. http://github.com/xtk/X. Accessed 11 June 2016
16. Node.js is a javascript runtime built on chrome's v8 javascript engine. http://nodejs.org. Accessed 11 June 2016
17. dat.GUI a lightweight graphical user interface for changing variables in Javascript. http://workshop.chromeexperiments.com/examples/gui/#1-Basic-Usage. Accessed 11 June 2016