VAST *Lite*
Volume Annotation and Segmentation Tool
User Manual, VAST Lite RC1

Daniel R. Berger

October 20, 2014

Contents

Chapter 1

Introduction

VAST is a utility program for manual annotation and segmentation of large volumetric (voxel) data sets. It enables users to work with voxel data sets in the Terabyte or even Petabyte range at interactive speeds, to explore them visually and to label structures of interest by voxel painting.

Voxel painting has a number of advantages over other manual segmentation approaches like bread crumbing (placing a number of labeled points inside each object) and skeletonization (placing labeled points and connecting them with edges to form a 'skeleton'). It reveals the true shape of objects and makes visualizations of the data more comprehensive. It also allows measurement of volume and surface shape properties of the labeled objects. When working on a dense segmentation, the fact that voxel painting labels areas rather than single points or lines in the cross-sections of objects makes it a lot easier to spot objects that have not been labeled yet (visual pop-out). It also allows for some additional functionality, for example reliably determining neighbor regions, using voxel overlap of several segmentations for merging regions or determining synaptic connectivity, and conditional painting. Such functions are also difficult to implement when using outline labeling (drawing vectorized lines around process cross-sections in 2D slices). In addition, outline labeling can introduce problems in identifying corresponding outlines in different slices and can cause region overlap. Last but not least, many machine learning algorithms for automatic segmentation rely on voxelized labelings for training, and the output of many such algorithms are labeled voxelized regions. VAST can be used to generate volumetric training data sets, and can to some extent also be used for importing, proof-reading and correcting results of segmentation algorithms.

The disadvantages of voxel painting which are mentioned most often are that it is slower than for example skeletonization or bread crumbing, and that it needs more storage space than alternative methods, especially when working with very large data sets. VAST tries to alleviate both disadvantages. For voxel painting, the time needed to label an object is largely determined by the number of outlines that have to be drawn. VAST can do automatic convex Z-filling to

reduce the number of outlines by a factor of up to 8 (for standard settings) if accurate boundary tracing is not needed. Painting is also optimized to be highly responsive, and the user interface provides quick access to functions which are used often (navigating through the image stack, changing the tooltip size, color picking, and switching between paint and delete mode). To reduce the amount of required memory and to enable interactive painting speed with any tooltip size, VAST implements multi-scale painting.

Considering that alternative labeling methods are likely to be more error-prone and will probably require more time until an acceptably low error rate is reached, voxel painting might be the overall faster alternative for fully manual labeling. If an object skeleton is needed (for example to compute length of a dendrite or number of spines etc.), it can be computed from the voxelized segmentation, whereas the inverse operation (computing accurate volumes from skeletons) is much harder. In a preliminary test we found that using VAST, a very experienced user can produce a dense segmentation of well-stained and well-aligned cortex neuropil data (6x6x30 nm voxel size) at a speed of about 4 cubic microns per hour. If cross-sections are labeled with a color dot in the middle rather than accurately outlining and filling the cross-sections, the labeling speed can be increased by a factor of 3 at the expense of accuracy (12 cubic microns per hour).

VAST is also very portable and light-weight. It consists of a single Windows executable file which is independent of third-party libraries. It does not require to be installed and can be easily copied and for example run from a memory stick.

Many tools for labeling voxel data exist, but such tools usually have drawbacks when it comes to painting in large data sets. Most tools do not support voxel painting but instead do bread crumbing, skeletonizing or storing object outlines as vector graphics (for example splines). Also most of them require the data to be loaded in RAM completely. This is not possible for the data sets in the Terabyte (and soon Petabyte) range which are currently produced by serial-section electron microscopy of biological samples. Many of these tools are also developed as cross-platform applications, so that they can be run on Windows, Mac and Linux systems. This usually means that a non-native GUI system has to be used (e.g. Qt) which makes the program a lot more bulky and more difficult to install and maintain. VAST is a Windows-only program and uses native Windows GUI and graphics functions.

The key concepts of VAST are:

- Image stacks are imported into VASTs .vsv file format, where they are stored as dices. Pre-computation of mipmaps allows for fast panning and zooming through the data when it is opened in VAST.

- .vsv files support lossless and lossy compression to reduce the resulting file size

- Image data and segmentations are stored in single files, which makes it easy to copy them from one place to another
- Support for loading EM stacks from a web server over HTTP (openconnecto.me format)
- Dynamic multi-threaded cacheing in RAM with pre-loading for low-latency display update
- Several image stacks can be opened and displayed together with a number of blending and tinting options
- Multi-scale painting (in VAST, painting always happens at the currently displayed resolution (Mip level))
- Automatic convex Z-filling during painting to speed up coarse labeling
- Automatic 2D-filling of closed contours
- Label color patterns to create a larger number of distinguishable label colors
- Label hierarchies for fast and reversible grouping of labeled segments
- Anchor points to quickly find a given segment in the volume
- Exporting of segmentations, EM stacks and mixed image stacks ('screenshots') in multiple formats
- Importing of segmentations from image stacks
- File-modification free editing. Image files are not changed (except if you explicitly update information in them). Segmentation files are only changed it you save the segmentation back to the same file.

Current limitations:

- Only one segmentation layer can be opened at a time
- Currently only 16-bit segmentations are supported
- There is no 'Undo' function
- There is no direct export function for 3D models of segmented objects (but this can be done by exporting the segmentation as an image stack and using a Matlab script to generate .OBJ files from it)
- No 3D model display
- No multi-user support
- Not an open-source project, but the VAST Lite executable can be copied and used freely (see section A.6 for details)

Please note that VAST is currently under development, and is subject to change as new features are added and bugs are fixed. For bug reports or helpful suggestions, please contact me at: `danielberger@fas.harvard.edu`.

Chapter 2

Getting Started

2.1 System Requirements

VAST currently only runs on 64-bit Windows computers that support DirectX 11. These are Windows Vista, Windows 7 and Windows 8 (all 64-bit), with DirectX 11 or later installed. Windows XP and older versions will not work. Currently only 64 bit versions of these operating systems are supported, because 32 bit programs are limited in the amount of RAM they can handle (4 GB max. theoretically, less realistically). The computer also has to have a DirectX 11 compatible graphics card. Luckily in most modern computers even the on-board graphics chips support DirectX 11.

Recommended system configuration:

- Windows PC with 64 bit Windows Vista, Windows 7 or Windows 8
- 16 GB of RAM (the more the better)
- DirectX 11 compatible graphics card
- 2 TB of disk space (depends on the size of the data you work with)
- Wacom Cintiq 13HD or other pen touch screen with two-button pen

Minimal system configuration:

- Windows PC with 64 bit Windows Vista
- 2 GB of RAM
- DirectX 11 compatible on-board graphics card
- Standard screen and mouse

2.2 Program Setup

To use VAST, simply copy the executable program into a folder where you have read/write access, and set up links on your desktop, start menu and/or taskbar if desired. It is important that VAST has read and write access to the folder where the executable is, because it will write a configuration file (`vast_preferences.dat`) into the same folder to store your settings.

Start the executable.

2.2.1 Try It Out!

The quickest way to have a quick look at VAST is to use an online data set. Several online data sets are included in the .ZIP package of supplementary files as .VSVR files. You can save this package from the executable by clicking 'Yes' in the 'First Start' pop-up window when you start VAST for the first time, or by choosing 'Save Documentation .ZIP To Disk...' from the 'Info' menu.

Unzip the package. Then, in VAST, go to 'Open EM...' and select one of the .VSVR files in the `VAST_package/Online Datasets/` folder, for example '`openconnectome_kasthuri11.vsvr`'. This will load images of a big EM stack from the Johns Hopkins `Openconnecto.me` server. Your computer has to have internet access for this to work.[1]

Click and drag the EM slice to pan. Use the mouse wheel to zoom. Use UP/DOWN arrow keys to scroll through the stack.

Click on the little pencil icon in the toolbar to switch to 'paint' mode. Choose 'Yes' in the popup window. Click and drag over the image to paint. You can select different paint colors in the 'Segment Colors' tool window. To erase, hold down the 'Delete' key while painting (or click and hold the right and left mouse buttons together). Select 'Keyboard Shortcuts' from the 'Window' menu for a list of available keyboard functions.

2.2.2 Preferences

In the main menu of VAST, go to '`File / Preferences ...`' to open the Preferences dialog window. Here you can set the parameters for data cacheing and display. VAST will set up the preferences for you when you run it for the first time on a computer (whenever it cannot find the preferences file). You can edit these preferences if you want to. You should at least check once whether the folder in which VAST puts its temporary disk cache is on a hard drive with lots of free space. Depending on what you do, temporary disk cache files can get similarly large as the segmentation files you are working with, in particular if you use global segmentation editing functions like merging, segment deleting, or segment welding.

[1] Remote EM images are loaded using the openconnecto.me cutout service over HTTP. The VSVR file just defines the web address of the data set to load and its dimensions.

Memory and Cacheing

On the left side you can set how much cache memory VAST will use maximally for voxel images and for segmentations. VAST chooses initial values which are reasonable for your system. The rule of thumb is: If you can afford it, leave 1-2 GB for the system, and split the rest 1/3 each for image cache, segmentation cache, and general usage of VAST (don't assign). On a system with 8 GB RAM, this means to give 2000 MB to the image RAM cache and 2000 MB to the segmentation RAM cache. If you are only viewing images and not using segmentations, you can increase the size of the image cache and reduce the size of the segmentation cache accordingly. If you plan to use other programs at the same time or run two instances of VAST, please reduce these values as needed. VAST will not immediately use all of the allotted memory, but it will stop reserving new memory for cache blocks and re-use old blocks when it reaches the limit.

In general, do not allow VAST to allocate more memory than the system has. This can result in severe performance issues. There is a memory usage indicator in the upper right corner of the VAST window which shows you how much memory is currently used. The blue frame indicates the maximum amount of RAM which VAST uses for image and segmentation cacheing. If the memory indicator becomes red and your system slows down, try to REDUCE the cache limits to allow Windows and VAST to use more RAM for other data.

Some of the segmentation cache is used for holding the currently displayed part of the segmentation in memory. When you exit the preferences dialog, VAST will tell you how much of the segmentation cache it needs for the current display settings and whether the cache size is sufficient.

'`Disk Cache Directory`': Here you can specify the folder where VAST stores its temporary disk cache. Click the '`[ ... ]`' button to browse. Set this to a folder where you have lots of free space (more than the size of the largest segmentation file you will be working with, since for certain functions VAST has to duplicate the segmentation data).

Painting

The segmentation bit depth is currently fixed at 16 bits. This allows for a maximum of 65535 labels; however, since each segment is represented in the tree view of the 'Segment Colors' window in VAST, memory limitations in the Windows system might prevent VAST from using that many labels.[2]

'`Tablet Mode (Pen Paints, Finger Moves)`': On some pen-enabled tablet computers VAST can distinguish finger and pen input. If this mode is enabled, the pen will paint and the finger will move the view when in Paint Mode.

[2]It runs fine with more than 6700 labels in one of our data sets.

Display Properties

'`Maximum Window Width`' specifies the width (or height, whichever is greater) of the largest window you will be using, in pixels. This value is used to determine how many textured tiles are needed to fill the entire window at all zoom levels. Setting this value smaller reduces memory consumption and increases cacheing speed, but if the value is too small, the image texture might not reach all the way to the sides of the window at all zoom levels.

'`Target Resolution Smaller Than`' lets you specify the effective resolution of the displayed textures on the screen in screen-pixels per texture pixel. This affects at what zoom levels which mipmaps are used. '`2`' is a good setting for this; '`1`' makes it more detailed but slower (and more memory-consuming), and '`4`' makes it faster but blurry.

'`Texture size m (texture is m^2):`' defines how large the texture tiles will be which are used for displaying image and segmentation textures. Depending on the graphics card some texture sizes might be faster than others. I recommend to leave this setting at 128.

'`Texture Smoothing`': You can set here whether you want to use texture interpolation. This reduces aliasing effects but can result in a slightly blurred appearance of the textures. The most natural setting of this is, in my opinion, '`All except Mip 0`', which will show pixels with sharp boundaries only if you zoom in more than the native resolution of the image data.

The remaining options are self-explanatory. Opacity values have to be set between 0 (fully transparent) and 255 (fully opaque).

Press `OK` after you're done configuring the preferences.

2.3 Additional Files Included With VAST

Under '`Info / Save Documentation .ZIP To Disk ...`' in the main menu you can save out additional files which are packaged into the VAST executable, as a ZIP file. Select a target location and save, then unzip the ZIP file.

Currently this includes a set of `.vsvr` files to access some large EM data sets remotely (see section 3.2.1), some Matlab scripts which can be useful for analyzing VAST data in Matlab, and this documentation as a PDF file.

Chapter 3

Working with VAST

VAST uses its own file format `.VSV` to store image data. It can open `.VSV` files immediately and navigate in them quickly. If your data is a stack of for example `.PNG` images, you will have to import it into VAST before you can use it. During importing the data will be saved into a VAST-specific `.VSV` data file, which allows quick access to arbitrary parts of the data. After opening a `.VSV` image file, you can create a segmentation by painting on top of the images, or you can open an associated segmentation file (`.VSS`) and view image and segmentation together. `VSS` files tend to get big quickly, but can be packed efficiently, for example in a `ZIP` file. You can view segmentations, modify and save them. You can export segmentations as image stacks for using them in other analysis programs or to render the segmented objects in a 3D animation program like 3D Studio MAX. You can also import segmentation image stacks that were generated externally.

3.1 Image Stack Importing

Typically volumetric image data is stored as a series of 2D images, or as a serial 3D block of data, which is not suitable for fast interactive viewing. When you import such a data set into VAST, it puts the images into a single file containing a diced data structure, and computes and includes mipmaps for the images.[1] Using diced data does not only speed up loading of parts of images, but will in the future also enable fast loading of volumetric sub-regions or 2D sections at other orientations through the image data.

VAST does currently not include image alignment and stitching functions. If you are starting with an unaligned stack of images, you will first have to align the images with a different program (Fiji or Photoshop, for example) and then save a stack of aligned images which all have the same dimensions and are named and

[1] A *mipmap* is a downsampled version of an image. VAST uses power-of-two (2D, XY) mipmaps. For example, for an original image of 1024x1024 pixels, it will compute mipmaps of 512x512 and 256x256 pixels. It does this for every slice image in a stack.

numbered in a consistent way (for example `img000.png, img001.png, ...`). Put all images into the same folder.

VAST can import single-tile image stacks, multi-tile image stacks, and 3D volume files. In a single-tile image stack, each slice of the stack consists of a single image file. In a multi-tile image stack, each slice is composed of several tiles in a XY grid, and each tile is stored in a separate image file. A 3D volume file stores all slice images in a single file. Currently the only 3D volume file format that VAST supports is NIfTI (.nii). VAST will convert image data to either 8-bit graylevel or 24-bit RGB when importing.

For importing and dicing, VAST will use the RAM cache which is normally used for cacheing EM image data during viewing and painting. Having lots of cache memory available will make importing somewhat faster, because images have to be re-loaded less often. You can set the size of the EM image cache in the Preferences (see section 2.2.2).

3.1.1 Importing image stacks: Pattern-based names

In the main menu of VAST, go to '`File / Import EM ...`'. VAST will show a file browser dialog in which you can select one or several image files. For importing 3D NIfTI files, please select only one file. If you import a single-tile stack and do not want to use pattern-based names, select all slice images in the correct order, because images will be stacked in the same order in which they appear in the system's list of selected files. The order is usually correct if you select the last image first, then shift-click (hold the `SHIFT` key down and click left with the mouse) the first image to select the whole range. You can also try 'Select All' by pressing `CTRL-A` if the folder only contains the image files you want to import. If you are worried about the order of the images and want more precise control, you can use pattern-based names. If you make use of pattern-based names to import single- or multi-tile image stacks, it is sufficient to select one file, but even better to select the first and the last file in your set of images. Then click '`Open`'.

After selecting one or more image files (not `.nii`), VAST will display the dialog shown in Figure 3.1. To import without pattern-based names, select '`Make Single-Tile Stack Using File Names and Order as Selected`' and press `OK`.

If you select the second option, '`Use Pattern-Based Names`', the parameters in the lower part of the dialog window will be enabled. With pattern-based names, you specify a template string for the file names which contains placeholders for numbers, and ranges for these numbers. With this you can also import image stacks in which each slice is stored in several image files (multi-tile image stacks). This is useful for data sets in which a single slice is so large that it can not be stored in a single image file, but is stored in a set of tiles which form a regular grid. Please note that these tiles should *not* be unstitched image tiles as they come off a microscope, but they have to fit seamlessly. If you have a set of raw microscopic images which are not yet stitched and aligned, please use an external program to generate a stitched and aligned image stack first, and store

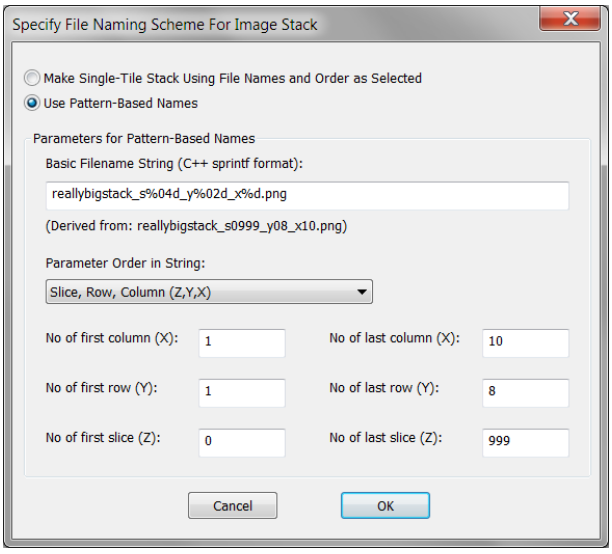

Figure 3.1: First dialog for importing EM image stacks: Specification of pattern-based names

each slice as a single image or a set of image tiles. You can then import those images into VAST.

VAST will use the file(s) you selected in the previous dialog to determine the source directory where the image files are and to generate a basic template for the file name. It assumes that all images of the stack are in the same folder (the one you picked an image from), and are named consistently with numbers for slices, rows and columns. It also assumes that the set of images is complete, which means that there's an image for every slice/row/column combination in the range you give. In this dialog, you specify these ranges as well as a schema to derive the filename for a given slice/row/column coordinate.

Let's say, for example, you have a data set called '`reallybigstack`', which has 1000 slices, numbered from 0 to 999, and each slice has 10x8 tiles, numbered from 1 to 10 and 1 to 8. You use a naming scheme so that the first image, in the upper left corner of the first slice, is called '`reallybigstack_s0000_x01_y01.png`', the image tile right of it is called '`reallybigstack_s0000_x02_y01.png`', and so on. The last image in the lower right corner of the last slice would be called '`reallybigstack_s0999_x10_y8.png`'.

First, make sure that the file name in the edit box at the top contains the correct *C++ format string* (as it is used by `printf()`). In general, numbers which specify the slice, column and row coordinates have to be replaced by codes like '`%d`' (integer number) or '`%04d`' (integer number with zero-padding to 4 digits). VAST will then fill in those numbers for each image. For more information about format strings, refer to a C++ manual or ask the internet.

In the combo box below, select which coordinates are used in the file names,

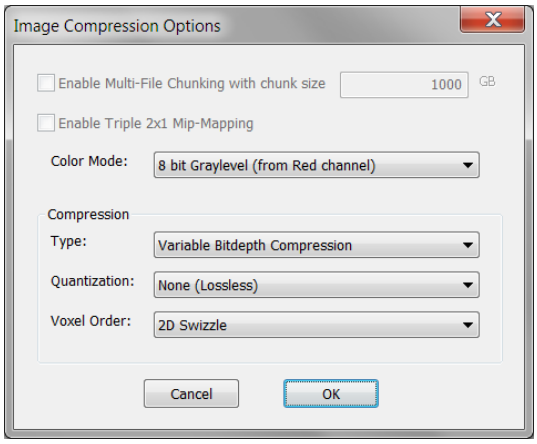

Figure 3.2: Second dialog for importing EM image stacks: Image compression options

and in which order they appear. The edit boxes below let you specify the range of (integer) numbers for the three coordinates. After you entered all parameters, press `OK`.

3.1.2 Lossless and lossy compression

Next, VAST will show a dialog where you can specify the color mode and image compression options (Figure 3.2). Multi-file chunking and triple mip mapping are currently not fully implemented and are therefore disabled.

Under 'Color Mode', please select if you want to import the images as 8-bit graylevel or 24-bit color images, and for graylevel which source color channel to use. When importing from graylevel images, please select the first option ('from RED channel').

Under 'Compression' you can specify compression options. Under 'Type' you can select the compression method - Uncompressed, Variable Bitdepth Compression, zlib Compression, and Spectral Compression. The three different compression algorithms are by themselves lossless, but might produce slightly smaller or larger file size depending on your data. Variable Bitdepth Compression should be fastest when reading from the compressed files.

'Quantization' specifies whether the compression should be lossy or lossless. Lossyness is achieved by quantizing, meaning throwing away bits. For example, if you set Quantization to -2 bits, graylevel images will have only 6 bits resolution (64 different gray levels) rather than 8 bits (256 different gray levels). Throwing bits away reduces both file size and image quality.

'Voxel Order' defines in which order the pixels in the images will be stored. This can have an effect on compressed file size. '2D Swizzle' stores pixels in 2D Z-order. I usually get best results using '2D Swizzle', but '3D Swizzle' might be

superior for very well aligned data (e.g. FIB-SEM).

Next VAST will ask you to specify a target location and file name for the resulting `.vsv` image volume file. Use '`.vsv`' as extension for the file name. Choose a location where you have enough storage space for the file. The file will not only contain the original image data, but also the mip maps. For example, if you import 1024 images of 1024x1024 pixels each and store uncompressed, the `.vsv` file will be approximately $1024 \cdot 1024 \cdot 1024 + 512 \cdot 512 \cdot 1024 + 256 \cdot 256 \cdot 1024 =$ 1409286144 Bytes ($\approx$ 1.3 Gigabytes) large. Lossless compression will reduce the file size, and lossy compression even more, but by how much depends strongly on your data and the compression method used.

Then, the images will be read, diced, and put into the target file. After that, VAST will compute the mipmaps and put those in the target file too. Depending on the size of the data, this process can take several hours. For example, a big data set of 350 GB takes about 5 hours to import on a recent desktop machine. The limiting factor is the speed of hard drive access.

You can cancel the importing, but the target file will then be incomplete / corrupted and can not be used with VAST.

3.1.3 Importing 3D volume files

Importing a 3D volume file is easier than importing an image stack. The only format currently supported by VAST is Nifti. VAST will ask you to specify the name of the source (`.nii`) file and the name of the target `.vsv` file. Currently VAST requires the whole Nifti file to be loaded at once into RAM, so this only works for smaller volumes. Also, the data in the Nifti file currently has to be 8 bit per pixel.

3.1.4 Image scale and description

After importing, you should set the voxel size of your data in the file. To do this, go to '`Info / Volume properties ...`' in the main menu. Here you can set the X,Y,Z size of a voxel in your data set in nanometers. Press 'Save to file' to store the information you entered in the VSV file. This dialog also displays how large your image stack is in voxels. The voxel size entered here is used for the scale bar which you can enable in the main menu under '`Info / View Scale Bar`'.

In the main menu under '`Info / EM File Information ...`' you can enter and view text which will be stored in your VSV file as well. This can contain a description of the data, copyright information, or other.

3.1.5 The RAM usage indicator

At the right side of the toolbar you can see a little field names 'RAM:' which shows the current RAM usage in your computer. The blue frame indicates how much RAM VAST will use maximally for segmentation- and image cache combined. Make sure that this frame is not dedicating more than 2/3 of your total

RAM (you can adjust these settings in the Preferences, see section 2.2.2). The solid blue block shows how much RAM VAST has currently allocated (including blocks allocated for segmentation and image cache). The green area shows you how much RAM the Windows system and other programs are using. The colors will change to yellow if the total memory usage goes above 90%, and red if they go above 96%. Running out of available RAM can slow down your system significantly. However, in some cases Windows uses large amounts of the available RAM for disk caching and can free those instantly if more RAM is needed by programs without affecting the system performance.

3.2 Viewing and Navigating an Image Stack

After you imported a stack of images, you can view them interactively. After you closed the program, you can re-open a previously diced data set by using '`File / Open EM ...`' from the main menu. VSV files you open will be added to a list under '`File / Open Recent EM`', from where you can quickly access them again. The list contains the 16 most recent VSV files.

You can also open VSV (and VSS) files by drag-and-drop from a file browser (Windows Explorer) onto the VAST window.

VAST currently has a 'Move mode', a 'Paint mode', a 'Collect Mode' and an 'Eyedropper Mode', which you can set by clicking the tool buttons in the toolbar. The cross of arrows icon selects 'Move mode' and the little pencil selects 'Paint mode'. In this section we will explain how to use the 'Move mode'. For an explanation of the other modes please refer to section 3.3.

The easiest way to navigate in the image stack is by using the mouse in 'Move mode'. You can pan (move the image sideways) by left clicking and dragging it. You can use the mouse wheel to zoom in and out. Alternatively, you can zoom using the N and M buttons, or the sidebar (see below). Use the UP and DOWN arrow keys or A and Z to scroll through the slices of the stack, or the sidebar to scroll more quickly.

3.2.1 Remote image stacks

In addition to using an image stack in VAST which has been imported into a local `.VSV` file, you can also open and access image stacks which are hosted online. VAST supports the *'Open Connectome Project Cutout Service'* from `http://www.openconnecto.me` with binary zipped data through HTTP; see: `http://www.openconnectomeproject.org/#!services/chru`.[2] Before you can access the remote image stack you have to generate a `.VSVR` file which specifies the parameters of the data set. `.VSVR` files are text files in a JSON-like format; here is the content of the `openconnectome_kasthuri11.vsvr` file:

[2]Essentially VAST requests [128x128x16] pixel blocks of the data set by reading from the Open Connectome server with URLs which specify the requested region, like: *http://openconnecto.me/ocp/ca/kasthuri11/zip/6/1,129/1,129/1,17/*. The received file is then unzipped to extract the image data.

```
{
  "Comment": "Source: http://openconnecto.me/ocp/ca/kasthuri11/info/",
  "ServerType": "openconnectome",
  "ServerName": "openconnecto.me",
  "ServerFolder": "/ocp/ca/kasthuri11",
  "SourceDataSizeX": 21504,
  "SourceDataSizeY": 26624,
  "SourceDataSizeZ": 1850,
  "TargetDataSizeX": 10747,
  "TargetDataSizeY": 12895,
  "TargetDataSizeZ": 1850,
  "OffsetX": 0,
  "OffsetY": 0,
  "OffsetZ": 0,
  "OffsetMip": 1,
  "TargetVoxelSizeXnm": 6,
  "TargetVoxelSizeYnm": 6,
  "TargetVoxelSizeZnm": 30,
  "TargetLayerName": "Kasthuri11@OpenConnectome"
}
```


VAST comes with a few pre-defined `.vsvr` files which you can use to open and view some example data sets.

3.2.2 The sidebar

VAST provides a sidebar for zooming and moving through the stack. The sidebar is a region close to the left and the right edge of the main window. When you move the mouse cursor to the left or right edge of the window you will see it appear as a transparent white overlay strip.[3] Clicking into the sidebar and dragging the mouse up or down will scroll through the slices of the stack (left mouse button) or zoom (right mouse button). If you move the mouse cursor too far away from the side of the window, the view will 'jump back' to the previous view. If you move the mouse cursor very close to the top or bottom of the window while scrolling (not zooming), VAST will start to scroll continuously, with a speed depending on mouse cursor position. You can use this function to quickly scroll through a very large image stack.

3.2.3 Getting and setting coordinates

VAST uses a coordinate system with a zero point in the upper left corner of the first slice, with positive X to the right and positive Y down in the slice, and Z marking the slice number. Coordinates are given in pixels at full resolution (the coordinates are independent of the mip map displayed). The coordinates displayed in the upper left corner of the main window show the current location of the center of the main window. You can switch the displayed coordinates on and off by using '`Info / View Coordinates`' from the main menu. Zooming in or out will not move the center point of the window and therefore also not change its coordinates. Getting or setting coordinates will also use the coordinates of

[3] You can set the opacity of the sidebar in the Preferences, under 'Side Bar Opacity'

the center of the screen, as do the 'anchor points' of segments (see section 3.3). While you drag the slice with the mouse VAST displays a transparent cross which indicates the location of the center.[4]

Once you load an image stack, a tool window labeled '`Coordinates`' will appear in the upper right corner of the main window. If the tool window is not displayed you can open it using '`Window / Coordinates`' from the main menu. It shows you the current center coordinates and allows you to read and set these values. The edit field in the tool window is updated as you navigate through the stack. To save the current location, simply copy the coordinates from that text field (mark with the mouse and press `CTRL-C`), then paste it into the text editor of your choice. You can also set the coordinates by entering or pasting numbers here and pressing `Enter`. VAST will then jump to the new coordinates. The exact format of the string does not matter; VAST simply looks for the first three numbers in the string. VAST does not mind whether there are commas or brackets or other non-numerical characters.

This function is quite useful if you want to store coordinates of interesting points in an external text file or spread sheet. Please keep in mind that the coordinate denotes the center of the current view. The center is indicated by transparent crosshairs when you pan the view. You can also center any point by right-clicking that point with the mouse in 'Move' mode and selecting '`Center`' from the context menu.

The dropdown-listbox in the `Coordinates` tool window lists the up to 64 most recent locations you visited. A new entry is added every time you pan the view (but currently not if you scroll through Z). You can go back to previous locations by selecting the coordinates from this list.

3.2.4 Layers

VAST can open several image stacks at the same time, provided that they have the same stack size. Each image stack, and also the segmentation stack, are listed as a 'Layer' in the 'Layers' tool window. For image stacks (not the segmentation) the order in the list defines the order of the layers. Layers BELOW in the list are 'in front'. The segmentation layer is always rendered on top of all image stack layers. You can change the order of the layers by drag-and-drop in the list. If you can not see all layers in the list, increase the size of the tool window by dragging a corner.

Below the list of layers, the 'Layers' tool window shows a number of 'Layer Properties' for the currently selected layer:

- '`Solo`': If this function is enabled, only the currently selected image layer will be displayed.
- '`Editable`': Disabled for image layers
- '`Visible`': Transparency value for this layer. Switch off to hide layer.

[4] You can set the opacity of the center cross in the Preferences, under 'Center Cross Opacity'

- 'Bright': Image Brightness; switch on to enable brightness control
- 'Contrast': Image Contrast; switch on to enable contrast control

VAST can blend layers with different transparency modes. Click on the button 'Menu' to access more layer options. The different settings for 'Blend Mode' are:

- 'Flat': All pixels in the image share the same transparency [Default]
- 'Dark Transparent': The darker a pixel ((R+G+B)/3), the more transparent it is
- 'Bright Transparent': The brighter a pixel ((R+G+B)/3), the more transparent it is
- 'Max(RGB) Dark Transparent': The darker a pixel (Max(R,G,B)), the more transparent it is
- 'Max(RGB) Bright Transparent': The brighter a pixel (Max(R,G,B)), the more transparent it is

'Color Filter ...' will open a color selection dialog where you can choose a color by which the layer images should be filtered during display. To not filter the images, choose white (255,255,255) [Default].

3.3 Painting

The main function currently provided by VAST is painting of segmentations as a colored overlay of the image data. When a stack of EM images is loaded, you can enter 'Paint Mode' by clicking the little pencil icon in the toolbar. When you start a new segmentation like this, VAST will ask you if you want to add 16 segments (label colors) to your segment list. Also, two floating tool windows will appear at the right side. The upper one, 'Drawing Properties' (Figure 3.3), provides options for drawing, whereas the lower, 'Segment Colors', lets you select and organize the segment labels and their colors in the segmentation.

When in paint mode, you can paint on top of the currently displayed EM image. Select a color (label number) from the 'Segment Colors' window at the right by clicking on it. Then click the left mouse button where you want to paint in the image.[5] You will see the outline of your current tooltip as a circle. By clicking and dragging the mouse you can paint larger regions. All painting happens in an overlay plane which is blended over the EM image (the EM image itself will not be changed). You can use the 'Alpha:' checkbox in the toolbar to switch the painted overlay on and off, and the slider right of it to set the opacity of the painted overlay. Most colors are not solid colors, but have patterns. Use the 'Pattern:' checkbox to switch patterns on and off, and use the slider right

[5] Even though you can use VAST with a mouse, it is designed to be used with a pen tablet.

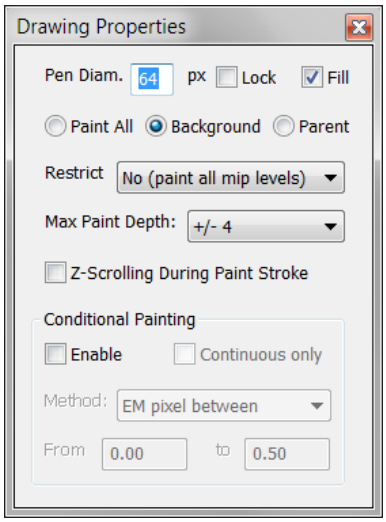

Figure 3.3: The Drawing Properties tool window

of it to manipulate the contrast of the patterns. If you enable the '`SelAlpha:`' checkbox, the opacity of the selected segment and its children will be controlled separately by the SelAlpha slider. You can use this to highlight a particular segment or set of segments. You can also switch the EM image layer on and off, by clicking on the '`EM`' checkbox in the toolbar. This is sometimes useful if you want to inspect just the segmentation.

You can change the size of the pen tooltip. The easiest way is, if you are using a pen tablet and VAST is properly configured, to hold down one of the pen buttons and to move the pen up or down on the screen. You can also use the - and + buttons. The current pen diameter is displayed in the Drawing Properties tool window. The third way of changing the tooltip size is to edit the '`Pen Diam.`' text field in the tool window. You can also lock the current tooltip size if you don't want it to be changed accidentally, for example if the size of the tooltip is important for your data analysis, by switching on '`Lock`'.

The checkbox '`Fill`' next to it switches automatic filling of closed contours on and off. If enabled, after each paint stroke VAST checks a rectangular area with the approximate extent of the stroke for empty closed contours of paint color and fills them with the paint color.

Below you can choose from '`Paint All`', '`Background`' and '`Parent`'. This determines which voxels in the current segmentation are paintable. If you select '`Paint All`', you will paint over or delete anything, no matter if it was painted before or not. If it is set to '`Background`', previous paints will not change, but your paint will only be applied to voxels which have not yet been painted to. If you erase, only the current paint color will be erased to empty (background). This is the most useful painting mode.

Instead of only affecting background pixels, '`Parent`' mode will affect only

pixels which have the color of the immediate parent of the current paint color (see below for a description of segment hierarchies). When erasing, voxels with the current paint color will be changed back to the parent color. This mode is only useful in special cases, in particular when re-labeling a previously painted area to a new color.

3.3.1 Multi-scale painting

A specialty of VAST is that it allows you to paint at different resolutions. In fact, VAST limits you to always paint at the currently displayed resolution. The advantage of this is that the amount of data that has to be manipulated when you paint a stroke is limited by the window size and screen resolution. Otherwise, for very large volumes one could easily get into a situation in which the amount of data that has to be written for a paint stroke is much larger than what can be loaded in RAM at one time, which would cause all sorts of problems, including very slow painting. Also it does not make sense to paint at a resolution which is much higher than the screen resolution because mouse (or pen) precision is also limited. Finally, allowing low-resolution painting can save a lot of memory, if large objects are painted coarsely.

In VAST, images are stored as a pyramid of mipmaps with reduced resolution using powers-of-two factors. Painting always happens at the resolution of the currently displayed mipmap. This means that you can change the resolution at which you are painting by zooming. A single segmentation can be composed of parts at different resolutions. For example it is possible to draw a rough outline of an object at a low resolution, and then to zoom in and correct the object's shape at a high resolution. VAST will automatically upscale and downscale the displayed segmentation as you zoom, but zooming will not change the painted segmentation. The segmentation is stored at the resolution at which it was painted. If you paint at a low resolution first and then correct at a high resolution, part of the low-resolution segmentation will be replaced by a high-resolution version. If you paint at a high resolution first and then correct at a low resolution, part of the high-resolution segmentation will be replaced by a low-resolution version, including pixels in the vicinity.[6]

Sometimes you might want to make sure that a painted segmentation has a certain resolution. You can enforce painting at only one resolution by restricting painting to a particular mipmap ('`Restrict`' in the `Drawing Properties` tool window). VAST will then enable painting only when the image stack is zoomed to display the selected mipmap.

3.3.2 Automatic Z-filling

The time that has to be spent to manually paint a segment in VAST depends largely on the number of 2D outlines that have to be drawn. Especially if you

[6] I have to do this because at the time of painting at a low resolution, not all higher-resolution images may be available in RAM (they may even be too large to be loaded in RAM).

follow a process that runs vertically through the volume, you have to paint (almost) the same outline over and over again, for every slice. If you want to just get a rough outline of an object and you're not interested in a high precision of the boundary, you could increase the painting speed by a factor of n if you paint the outline only in every n-th slice, or paint n slices at a time. In the first case, you get gaps of $n-1$ slices between the painted outlines, in the second case it is hard to determine what you are actually painting because you can't see where your color goes in most of the slices.

VAST uses a third method. It supports automatic Z-filling of intermediate slices where the regions of the lower and the upper painted region overlap. It turns out that in most cases neuronal objects are *locally convex*. Exceptions are branches, for example when a spine neck runs very close to the dendritic shaft. Automatic z-filling will only fill in the volumes of the overlap between the specified painted regions, and in most cases (for convex objects) the filled regions will stay inside the segmented object.

Z-filling makes sense across a few slices only, because there will be no overlap if your object moves too much from slice to slice (runs oblique). Also, VAST has to load multiple slices in RAM to be able to fill in those slices. The maximal distance across which VAST lets you fill in depends on the size of the image cubes used. Currently the cubes are set to be 16^3 voxels large, and VAST allows you to fill in up to +-8 slices (because it loads two layers of cubes at a time). In the data sets we are using this is approximately as far as z-filling makes sense, and it speeds up painting by a factor of 8.

You can set how far the z-filling will reach by setting '`Max Paint Depth`' in the `Drawing Properties` tool window. This value controls both the distance at which Z-filling occurs, and the stepping distance for navigating with S, X or PageUp, PageDown keys, to ensure gap-free painting.

Automatic Z-filling is only applied while painting, not when erasing. This makes it easier to correct what has been filled in in the case of non-convex neighborhoods. This also means that the best strategy to draw an object coarse-to-fine is to try to paint conservatively (try to stay within the object boundaries), and correct by adding paint rather than removing paint, because deleting has to be done in every slice individually.

'`Z-Scrolling During Paint Stroke`' is by default disabled to prevent painting errors when accidentally switching to the next slice before the paint stroke is finished. However, if you enable it, you can very quickly coarsely label a long neurite running through your stack vertically by scrolling through the stack while following the neurite with your pen – provided that loading of the image stack keeps up with the update rate of the screen.

3.3.3 Using conditional painting

The last section of the `Drawing Properties` tool window handles the settings for 'conditional painting'. If you switch on conditional painting by clicking the check box `Enable`, only pixels will be painted for which the EM image fulfills certain criteria. You can choose from three methods which determine

the paintable pixels depending on whether the (normalized) brightness of the image pixel (of the selected image layer) is in a certain range, which you can set. The value range for minimum and maximum brightness is 0..1. If you are using several image layers and conditional painting does not seem to work, please make sure that the correct image layer is selected in the 'Layers' window.

Currently conditional painting does not work very well with z-filling. I recommend not using z-filling (set it to 0) when you use conditional painting.

3.4 Segments

3.4.1 Picking segments

You can select the segment color to paint with in the '`Segment Colors`' tool window by clicking on it in the tree view. You can also pick any color you see in the segmentation layer by using the pipette tool. To do this hold down the `SHIFT` key and click on the segment you wish to select. This makes is very easy to switch between segment colors while painting. Alternatively you can use the Pipette mode which you can select in the main toolbar. If you hover over segment colors in the main window when in picking mode, VAST will display the name of the segment as a tooltip.

3.4.2 The segment hierarchy

VAST can arrange segments in a tree-like hierarchy. This means that each segment can have other segments as children, which can themselves have children, and so on. VAST also allows you to collapse and expand parts of the tree dynamically, so that you can quickly switch between a visualization which shows a whole branch of the tree in the same color or individual sub-branches in individual colors. For example, if all spines of a spiny dendrite are labeled as sub-objects (children) of the dendritic shaft, one can instantly flip between a display in which the whole dendrite has the same color, or each spine has a different color, by opening and closing the dendritic shaft folder. Segments can also be used as folders to group segments, for example to classify labeled objects. You can use tags to designate certain segments as folders, to help external analysis (see section 3.4.10). The grouping can also be applied when segmentations are exported.

Segment hierarchies are visualized and edited in the 'Segment Colors' tool window. This tool window uses a 'tree view control', similar to the navigation pane of a windows explorer window, which makes usage very intuitive. Most advanced functions can be found in the tool window's menu, which opens either by clicking the 'Menu' button or right-clicking into the tool window.

3.4.3 Re-ordering and moving segments in the tree

To re-order the segments, simply drag and drop them with the mouse. You can only select and drag one segment at a time, but if the segment has children

the whole branch will be moved (including all children). Please note that to make a segment the first child of another segment, you have to drag it to the right side of the tool window, right of the new parent segment. The new parent segment will then be highlighted in blue, instead of the black line indicating the target space between two segments. You can move any segment, with exception of the 'Background' segment. The Background segment can also not have any children.

Because it can be cumbersome to move hundreds of items from one folder to another one by one, VAST currently supports two functions 'Make all siblings children' and 'Make all children siblings' which can help in certain situations (see section 3.4.7).

3.4.4 Collapsing and expanding tree branches

You can collapse and expand tree branches, which are displayed in the same way as folders and subfolders are in the Windows explorer, by clicking on the little '+' or '-' sign left of parent segments. When you collapse a folder, in the segmentation layer all its children will be displayed in the same color as the parent. If you pick a segment color from the segmentation layer by shift-clicking, and the selected segment is in a collapsed folder, the folder will be automatically expanded to show the native color of the segment you selected.

3.4.5 Using anchor points

Each segment has an 'Anchor Point' stored with it. This is an XYZ coordinate vector which indicates the location of the segment in the stack. Initially the anchor point is set to the point at which the segment is painted first. You can jump to the anchor point by right-clicking on a used segment in the '`Segment Colors`' tool window and selecting '`Go To Anchor Point`' from the context menu. You can quickly jump to the anchor point of the selected segment by pressing the 'Home' key.[7] You can also set the anchor point of the selected segment to the current view location (as indicated by the center cross) by selecting '`Set Anchor Point`' from the context menu. You will have to confirm this action in a pop-up window to prevent accidental setting of anchor points.

3.4.6 Adding new segments

There are several functions to add segments in the context menu of the 'Segment Colors' tool window. You can add a segment as next sibling or as a child of the selected segment. '`Add 10 Segments`' will add 10 segments immediately after the selected segment. '`Add Skeleton Segments`' adds a set of child segments to the selected segment which can be used for rudimentary skeletonization. I have not found this function particularly useful though.

[7] If you pressed the 'Home' key accidentally and want to go back to where you were, you can select the previous location from the drop-down menu in the 'Coordinates' tool window.

The most sophisticated way to add segments is '`Add Named Segments ...`', which lets you specify a naming scheme and add multiple named segments at the desired target location in the segment tree. VAST will attempt to guess a naming scheme from the name of the currently selected segment.

You can change the name of any segment in the same way as file names are changed in the Windows Explorer – click a selected segment name a second time, then rename it.

3.4.7 Helper functions for arranging segments

Under '`Arrange`' in the context menu you can find two useful functions to move many segments at once. '`Make All Siblings Children`' will move all siblings of the selected segment into its folder (make them children of the selected segment). '`Make All Children Siblings`' moves all children of the selected segment out of its folder and makes them siblings. Be careful with these functions because currently there is no 'Undo'.

3.4.8 Select recently selected segments

Under '`Select Recently Selected`' in the '`Segment Colors`' tool window's context menu you can find a list of the segments you had recently selected. You can click on one of the listed segments to select it again.

3.4.9 Global operations: Deleting and welding segment subtrees

Deleting and welding segments is actually more difficult than it seems because it involves traversing the whole segmentation data set and inspecting every single painted voxel. When you choose to delete the selected segment and its children, VAST will actually have to not only set all voxels with those segment numbers to 0, but also renumber all the other voxels so that the used label numbers will have no 'gaps' (all label numbers between 0 and n are used).

'Welding' will make the selected segment and all its children the same label. It will renumber all voxels with label numbers of children of the selected label to the same number as the selected label, and also, similar to when deleting, renumber the other segment voxels so that the resulting segmentation is free of label number gaps.

Because these functions change almost the complete segmentation and VAST is designed so that the opened segmentation file is not changed, it basically has to copy almost the complete segmentation data to the temporary cache file. Depending on what segmentation you are working on this can take a lot of time, RAM and disk space. Also it will change the internal ID numbers of the segments, so please think twice before using these functions in case you are relying on absolute segment IDs in your analysis. These functions are also not well tested; please report any bugs you might encounter.

3.4.10 Segment tags

Each segment can have a 'Tag' which you can select in the '`Segment Colors`' context menu under '`Tags ...`'. A tag is a number between 0 and 15 which can indicate the type of the segment. The tag value is exported with the Segment Colors text file and can be used to help external analysis. By default the tag of all segments is 0. VAST uses tag 1 to indicate that the segment is a 'Folder Segment', which is not a labeled object by itself but rather a folder which contains other folders and objects. This information can be used to collapse all objects, but expand all folders – select '`Expand Only Segments Tagged as Folder`' under '`Expand / Collapse Child Folders`' in the '`Segment Colors`' tool window context menu.

Segments which have a tag that is not 0 will have an icon with a colored frame in the Segment Colors treeview. Folder segments have a gray frame.

3.4.11 Editing the color of a segment

The color and pattern of any segment can be changed. Right-click on a segment and go to the sub-menu '`Colors`' of the context menu, then choose the desired option. Each segment has a primary color, a secondary color, and a pattern that is used to blend between them. You can also randomize the colors of *all* segments or set the primary or secondary color of *all* segments to the same color (not recommended). Please remember that by collapsing segment folders you can quickly and reversibly switch the displayed color of a segment to the color of the collapsed parent.

3.4.12 Exporting segment metadata

The entry '`Save Segment Colors...`' in the context menu lets you export the metadata of the segments to a text file, which you can then for example parse with MATLAB to extract colors, hierarchies, names, anchor points etc. of the segments for analysis.

3.4.13 Segment information

'`Segment Info`' in the context menu opens a window which shows you the internal information associated with the selected segment. You can use this information to count children of the segment, get its internal ID or other parameters. The text can be copy-pasted if needed.

3.4.14 Searching for a segment with a given name or ID

At the top of the '`Segment Colors`' tool window is an edit field which can be used to find segments. As you type or paste a text string into this field, VAST will select the next segment (after the selected segment in depth-first search order) the name of which contains the typed sub-string. The edit field

is case-sensitive. If there are more than one segment which contain this sub-string, you can click on the magnifying glass right of the text edit field to go to the next segment that matches your text. The F3 key has the same function, provided that the segment tree sub-window of the '`Segment Colors`' tool window is active.

You can also search for a segment with a particular ID. To do this, type an opening bracket `[` into the find edit field, followed by the ID number you are looking for.

3.4.15 The 'Collect' tool

The 'Collect' tool ('Collect Segment Mode') can be selected in the toolbar by clicking the icon with an arrow pointing at a folder. When in Collect mode, segments you click will be moved in the segment tree to become children of the currently selected segment. This can be useful to quickly classify different objects into different types. Simply make a folder segment for each type, select it, and click on the objects in the image that are of that type with the 'Collect' tool.

Using this tool is a bit dangerous because there is no Undo. If you click the wrong object it might be difficult to remember where it came from and there is currently no easy way to 'move it back'. Secondly, when an segment is moved, all its child segments are moved with it, but not its parent(s). So if you are dealing with objects which consist of several parts, make sure that you move the parent segment of the object and not only a side branch of its tree.

3.5 Saving Segmentations

Important: VAST DOES NOT SAVE AUTOMATICALLY while you paint. Your tracings will be held in RAM and/or a cache file on disk until you explicitly save them. If you open a segmentation from a `.VSS` file and work on it, the file will not be changed unless you explicitly tell VAST to save the changes you made to the opened file by selecting '`Save Segmentation`' from the main menu. If you want to keep the previous version and save to a new file, use '`Save Segmentation As ...`' instead. VAST will then take all data from the opened file, the RAM cache, and the segmentation cache file, and combine them into a new file on disk.

We have had cases in which people had VAST open for several days without saving and lost a lot of work when the computer crashed. Please save your work once in a while.

3.5.1 Save Segmentation As Special

'`Save Segmentation As Special ...`' provides you with two functions to save your current segmentation to a new file in a modified way. First, you can choose to save only the selected segment or subtree of the current segmentation to a

new file. Alternatively, you can change the resolution of your segmentation and adjust the canvas size on which the saved segmentation will open (this is currently only supported when saving all segments).

To save only the selected segment or the selected segment and its child tree to a new segmentation, select that option from the drop-down list and press 'Save'. Please be aware that the internal IDs of the segments in the new segmentation file will be changed to maintain a gap-free list of IDs from 0 to n for n segments.

The settings in the lower part of the dialog provide limited functionality to adjust the resolution of the saved segmentation and target canvas size.

Normally VSS files only open on top of an image stack of the same size in voxels. You can use this function to save an existing segmentation so that you can open it on a stack which has slightly different size and/or is scaled up or down by a power of two.

Currently the saved segmentation will stay aligned to the upper, left, top corner of the stack. Also, only powers-of-two scaling is possible. If you want to translate the segmentation to a different location or scale in the target stack, you'll have to export the segmentation as an image stack, modify the images accordingly, and re-import.

3.6 Segmentation Merging

If you have two or more (for example partial) segmentations (`.VSS` files) of the same image stack you can merge them into a single segmentation. For this, select '`File / Merge Segmentations in ...`' from the main menu, and choose the `.VSS` file(s) you would like to merge in with the currently opened segmentation. During merging, VAST will add the selected `.VSS` files to the current segmentation as if someone painted them on top in 'Paint All' mode. You can select several `.VSS` files at once for merging, which will then be added one-by-one. Segment IDs of the files which are merged in will be changed so that they do not overlap with existing segments.

VAST does not save the merged segmentation automatically nor does it change any of the segmentation files. It will generate the merged segmentation in the segmentation cache, and if it does not fit in RAM it will end up in the temporary segmentation file. You will have to save the resulting merged segmentation if you want to keep it, for example via '`File / Save Segmentation As ...`' from the main menu.

3.7 Importing Segmentations From Image Stacks

VAST has some – not very well tested – functionality to import segmentations from image stacks. Just like the image files generated during segmentation exporting (see section 3.8 below), the RGB values of the image pixels should encode the segment numbers (the least significant 8 bits (0..7) are in the blue

channel, bits 8..15 in the green channel, and bits 16..23 in the red channel). Please remember that VAST can currently only handle 16-bit segmentations.

To import, select '`File / Import Segmentation ...`' from the main menu. VAST will ask you to select one or several image files of the image stack you want to import. Then it will open a dialog where you can specify the parameters for segmentation importing.

File name(s):
You have to specify a name template for all image files in the stack (see section for more information on the format of the string). You can also specify the order and the value limits for the parameters of your file name template, to define the names of all the images or image tiles in your stack.

Image Parameters:
Here you can rotate and flip the images if necessary, and tell VAST where to put them into the currently opened volume. The 'Tile Size' is extracted from the image file you selected, but you can adjust it here too.

Segment Label Parameters:
If you have a segment metadata file for the imported segmentation stack (same format as the file written in section 3.4.12) you can provide it here. The options below specify how VAST should deal with the segment IDs in the imported stack.[8]

Similar to segmentation merging, VAST will not save the segmentation after importing automatically. You can do that yourself after importing, using '`File / Save Segmentation As ...`' from the main menu.

3.8 Exporting Image Stacks

Figure 3.4 shows how we currently analyze and visualize manual segmentations done in VAST. Segmentations can be exported as image stacks, and these image stacks can then be imported into MATLAB, where we compute properties of the segments and generate mesh surfaces. MATLAB can export meshed surfaces to the common .obj file format, which we load into 3D Studio MAX for visualization.

Go to '`File / Export ...`' in the main menu to export (parts of) the image stack and/or segmentation stack as a stack of image files. The dialog shown in Figure 3.5 will pop up. You can export EM image stacks, segmentation image stacks, and screenshot image stacks. You can specify a region of the data set to be exported, and a resolution (mip level) for the images. You can also export a data region which is too large for storing the whole slice in a single image as a tiled set of images.

[8] Not all options might work. Some restrictions apply.

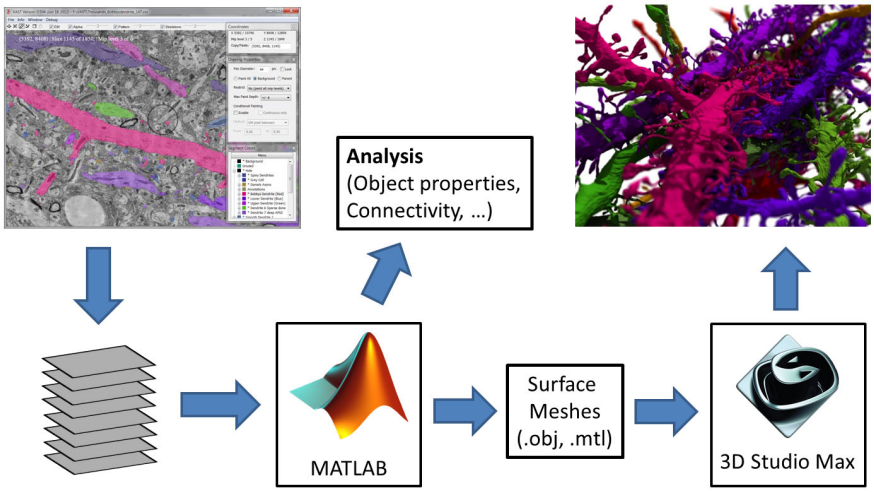

Figure 3.4: Current pipeline for exporting from VAST

Export as:
Choose here if you want to export an image stack as a single-tile stack (one tile per slice) or a multi-tile stack (a grid of image tiles per slice). For the second option you can define the tile size to be used.

Region to export (Specify coordinates at full resolution):
This defines the region of the stack that should be exported, for all three targets (Screenshot, EM (image) data, Segmentation). By default it is set to the full stack. You can restrict the export region here. X is the horizontal axis in the slice, pointing rightwards; Y is the vertical axis in the slice, pointing downwards on the screen, and Slice (Z) defines the range of slices that should be exported. The first and second columns of the edit fields let you define start and end of the region for each axis, the third column defines the size of the exported region (edit fields change each other to stay consistent).

If you click the '`Full`' button, the values will be set back to the full extent of your stack.

The '`BBox`' button sets the region to the bounding box of the currently selected segment. You can use this function to define a cut-out region from a painted segment;[9] you can also use a new segment and just paint the upper left top and lower right bottom corners of the cutout region, select it and go to 'Export Data'. Then click the '`BBox`' button to use the painted extent as an export region.

Under '`Resolution:`' you can select at which mip level you want to export.

[9]Note that the bounding box is not always correct, in particular if you delete parts of what was painted before, the bounding box will not shrink.

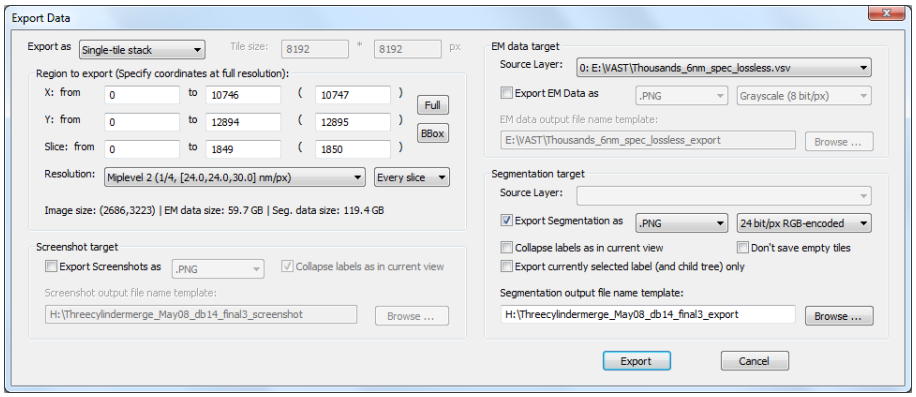

Figure 3.5: Export dialog

VAST does not support arbitrary scaling, but can export image stacks at its native mipmap scales (which are powers of two). You can also subsample the stack by slices (every nth slice).

Below you can see the image size resulting from your settings and an estimate of the (raw) data size that will be exported. Compressed image formats like .PNG can however produce much smaller file sizes, depending on the image content.

Screenshot target:
If you want to export a stack of 'screenshots' how the images look in the main window of VAST, enable the checkbox '`Export Screenshot as`'. VAST will reproduce the pattern, blending and tinting settings as they are currently set in the main window in the exported 'screenshots'. Select a target image format and filename prefix / location.

EM data target:
This saves a stack of (EM) image data from the selected layer. You can specify the target format, filename prefix and location.

Segmentation target:
This saves the segment IDs of the segmentation layer or part of it as an image stack. The segment ID of each pixel (a 16-bit number) will be encoded in the color of the pixel in the exported image. Bits 0-7 will end up in the blue channel and bits 8-15 will end up in the green channel. The red channel will currently stay 0.

When you are done setting up the parameters for the export, press '`Export`' and VAST will start exporting the image stack or stacks (VAST can export more than one target at the same time).

Appendix A

FAQ and Trouble Shooting

A.1 Frequently Asked Questions

My image stack is not aligned. How do I get it aligned into VAST?
VAST does not have any stack alignment (not stitching) functionality. You'll have to use other programs to render an aligned image stack (for example Adobe Photoshop or plugins in Fiji [2], [4]), and then import that aligned stack into VAST; or you'll have to work on an unaligned image stack.

Can I analyze multi-channel optical image stacks in VAST?
Yes. You can load several image stacks at once, provided they are the aligned and the same size, and each one can be RGB or graylevel. In addition you can tint different image stacks in different colors to distinguish different channels.

Does VAST support 4-dimensional data sets (for example time-lapse data of a 3D structure)?
No.

How do I open a .VSS file without a matching .VSV file in VAST?
You can not. Actually, you can, with a work-around. If you load any `.VSV` and then open the `.VSS` in question, and the `.VSS` file has a different size, VAST will tell you its dimensions (size in pixels) in the error message window. Note down these numbers. Now create a dummy `.VSVR` with these dimensions (`VSVR` files are just text (.txt) files with a different file name extension). Set the field 'ServerName' to an empty string to make it a dummy layer. Then restart VAST, open the dummy `.VSVR` as image stack, and open your `.VSS` (which should have the same size) on top. Here is an 'empty dummy' `.VSVR` example:

```
{
  "Comment": "Empty Dummy Layer",
  "ServerType": "openconnectome",
  "ServerName": "",
```

```
  "ServerFolder": "",
  "SourceDataSizeX": 49152,
  "SourceDataSizeY": 32768,
  "SourceDataSizeZ": 255,
  "TargetDataSizeX": 49152,
  "TargetDataSizeY": 32768,
  "TargetDataSizeZ": 255,
  "OffsetX": 0,
  "OffsetY": 0,
  "OffsetZ": 0,
  "OffsetMip": 0,
  "TargetVoxelSizeXnm": 6,
  "TargetVoxelSizeYnm": 6,
  "TargetVoxelSizeZnm": 30,
  "TargetLayerName": "Empty Dummy Layer"
}
```


How do I deal with self-touching objects?
If you need to be able to recover the true shape of an object, for example for correct skeletonization or computation of the surface area, places where there are self-touches (for example, a dendritic spine touching the dendritic shaft) can be problematic. One way to get around this is by using sub-objects and glue. Just like a plastic model which is constructed from parts, you would make the spine a child of the parent and add 'glue' – a different segment which you treat specially in the analysis – to the interface where parent and child object are actually connected.

How do I make shiny 3D pictures and animations from the segmentations I painted in VAST?
VAST does currently not have 3D rendering capabilities. We use Matlab scripts to extract `.OBJ` (Wavefront OBJ) model files from exported segmentation image stacks (see section 3.8). These models can then be loaded into 3D rendering programs (we use Autodesk 3dsMax).
Please contact me at `danielberger@fas.harvard.edu` if you are interested in these scripts.

Suddenly all internal segment IDs changed – What happened?
VAST currently keeps a continuous list of segment IDs between 1 and n for n segments at all times. This means that if segments are removed from a segmentation, the other segments will 'move up' to keep the list continuous. This happens when you use the functions '`Delete Segment + Subtree`', '`Weld Segment Subtree`' or save a new `.vss` file with a subset of the current segments using the '`Save Segmentation As Special ...` function. If you are using the internal IDs to identify particular segments and do not want them to change, avoid those functions. For example you can put deleted segments into a 'Deleted' folder and/or re-use them. If you use '`Merge Segmentations in...`' or with certain settings '`Import Segmentation ...`', their internal segment IDs will also be changed so that they don't overlap with existing IDs.

Why is it called 'VAST *Lite*' and not just 'VAST'?
The 'Lite' in the name emphasizes that this is not supposed to be the final version of the software. It is currently a usable tool with a limited set of capabilities, which is provided to the scientific community without restrictions. Development of VAST continues, and there may at some point be a released version with more features.

A.2 Typical Use Cases

Analyzing synaptic connectivity in a cortex EM stack
It is possible to define the location of synapses and their synaptic partners by painting the synaptic membrane. We do this in two steps. First, we paint the axons and dendrites with individual colors. Then we save this data set, and generate a second data set in which just the synapse membranes are labeled, or in which the synapse membranes are over-painted with a special synapse color, with a pen with fixed size (for example 16 pixels diameter). Make sure that the 3D region painted for each synapse is a single connected component and that different synapses are separate connected components. We then export both data sets as image stacks and use a Matlab script to find each synapse by doing connected-component analysis. For each synapse we then find the axon and dendrite which occupy the same voxels as the synapse in the other data set, which gives us the connectivity information.

Counting and classifying objects by painting
Just as for the synapses in the previous example, you can use connected-component analysis in Matlab to count other objects in the stack, for example neuron cell bodies. You can use paint all neurons of one type in the same color and use different colors for different types. Connected-component analysis can be used to separate the different cell bodies for each type, given that they are separated in space. The connected-component analysis will also give you the number of objects of each type, and their volume, if you count the painted voxels for each connected component.

Tracking objects in a video
If you translate a video into a sequence of images, you can of course import this image sequence as a stack into VAST (even in color). In the same way as you can label three-dimensional structures in VAST, you can label objects or regions or fiducial points as they move through the video. You can then export the labelings as an image stack and analyze locations in the image and movement.

Defining fiducial points in an unaligned image stack for manually aided alignment
Some EM image stacks are difficult to align with automatic methods, for example if the image quality is bad, there is high-contrast background, or the tissue

slices have folds. Manually defined fiducial points which should end up in the same place from slice to slice can help improve the alignment. You can load an unaligned stack into VAST and use manual painting to define fiducial points. Use a different color for each feature you are tracking through the stack, so that in the analysis you know which points belong together.

A.3 Some Performance Tips

- If file access is very slow (when you move through the stack and it takes time until the images appear) consider storing the data locally and/or on SSD drives. In particular, especially when using non-SSD drives, put files which you use together on physically separate drives. The problem is often that two files (for example an EM layer and the segmentation layer) are loaded at the same time from the same non-SSD hard drive, which causes the read/write head to jump forth and back between two locations at high speed. This slows down file access a lot.

- If you experience a low frame rate (the mouse cursor is jumping rather than moving smoothly), try to reduce the size of the 'Maximal Window Width' in the Preferences to something like 1280. On very large screens you can set the 'Target resolution smaller than' to 4 (Default: 2) to help.

- If VAST slows down considerably after using it for a while, check if the RAM of your computer is full (check the RAM usage indicator in the upper right corner of the VAST main window, see section 3.1.5). Once memory is full, Windows might swap parts of the data to the hard drive, which can slow down processing a lot. To fix that problem tell VAST to use less memory by REDUCING the maximal RAM cache memory sizes in the Preferences dialog.

A.4 Setting up VAST with a Wacom screen

When it comes to fast and accurate manual painting on a computer, tablets and in particular tablet screens can improve productivity significantly. We are using various Wacom Cintiq screen tablets. While the larger Cintiqs have a better screen, they can be expensive and bulky. I personally prefer the Cintiq 13HD, which can be laid flat onto a desk and close to the user.

An alternative could be tablet laptops, if they fulfill the system requirements for VAST. We tried the Asus EEE Slate EP121; it works but it is a bit slow, and pen button presses tend to be unreliable. Newer tablets like the Microsoft Surface Pro might have improved performance, but some might not support two-button pens.

Wacom tablets come with driver software which lets you configure the pen buttons for each program. For optimal workflow in VAST I find it most useful to have 'erase' on one pen button and 'change tooltip size' on the other. For

this it is easiest to set one pen button to 'right click' and the other one to 'middle click'. If that does not work for your system (sometimes Windows uses the Right Click event in a special way, for example), you can configure the pen buttons to simulate equivalent key presses (see Appendix A.5).

If you see brief circular animations when you use the pen and drawing is delayed, you should go to 'Pen and Touch' in the Windows Control Panel and switch off 'Flicks'. On the tab 'Flicks', uncheck 'Use flicks to perform common actions quickly and easily' and click 'Apply'.

A.5 Keyboard Shortcuts in VAST

You can open a window which lists all the keyboard shortcuts in VAST from the main menu under `Window / Keyboard Shortcuts`. Here is a summary.

`SHIFT or ENTER`	Pick segment color by mouse click
`CTRL or INSERT`	Temporarily go to 'move' mode
`TAB or |`	Move mode: Click left and move pen up/down to zoom;
	Paint mode: Click left and move pen up/down to change pen diameter
`‘~ or DELETE`	Paint mode: Erase mode
`H or L`	Hides segmentation while held down

Table A.1: Mode Modifiers (hold down)

`UP or A`	One slice up
`DOWN or Z`	One slice down
`PAGEUP or S`	`MAXPAINTDEPTH` slices up
`PAGEDOWN or X`	`MAXPAINTDEPTH` slices down

Table A.2: Slice Navigation

`MAXPAINTDEPTH` is the value set under 'Max Paint Depth' in the 'Drawing Properties' dialog.

`-_`	Decrease tooltip diameter
`=+`	Increase tooltip diameter
`N or Keypad /`	Zoom out
`M or Keypad *`	Zoom in
`F`	Flash selected segment
`I, O, P`	Set paint mode to Paint all, Background, Parent
`HOME`	Go to anchor point of selected segment
`,< / .>`	Select previous / next segment in recently selected list

Table A.3: Other Controls

A.6 Terms of Usage and Privacy Statement

This version of VAST ('the software') is free of charge and may be distributed freely, but not sold. Commercial usage is allowed.

You are using this software at your own risk. Even though it has been tested extensively, it is not free of bugs. Please keep backup copies of your data.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

VAST does not collect usage statistics or other data. In particular, it does not transmit any of your image or segmentation data anywhere.

This software uses `easyzlib.c`, which is based on the `zlib` library by Jean-loup Gailly and Mark Adler.

Appendix B

Technical Information

B.1 Size limitations

Maximal file size for EM images and segmentations: Limited by maximal file size on disk; theoretical maximum 2^{64} bytes.
Largest EM stack that has been imported into VAST so far: ~ 4 Terabytes

Maximal number of labels supported: currently $2^{16} - 1$ (but needs lots of RAM; 600 bytes per label).

Largest image supported (at full resolution): $(2^{31} - 1) \cdot (2^{31} - 1)$

B.2 Supported file formats for importing / exporting

Importing of EM images

Currently 8 bit graylevel and 24-bit RGB image stacks are supported.

Stacks and tiled stacks: .JPG, .PNG, .TIF
3D Volume files: .NII (NIfTI); will be converted to 8 bits when imported

Importing from .TIF images under Windows 7 can result in an image stack that has only 16 gray levels instead of 256, because of a bug in the Windows GDI+ TIFF routines. Consider converting the TIFF images to PNG before importing.

Exporting of EM images

Stacks and tiled stacks: .PNG 8-bit indexed, .TIF uncompressed, .RAW

Importing of segmentations

Segmentations can be imported from RGB .TIF and .PNG image stacks. The segment number for each pixel is encoded in the RGB value of the image as

follows: Bits 0-7 of the label number are expected in the blue channel, bits 8-15 in the green channel, and bits 16-23 in the red channel. This is the same format used for exporting segmentations to image stacks (see below). Please be aware that VAST can currently only handle 16-bit segmentations.

Exporting of segmentations

When exporting segmentations, the available file formats depend on the range of segment numbers used. For example, if the highest segment number is greater than 255, 8 bit indexed file formats will not be available. In that case the label numbers will be encoded into the color channels (for example for RGB, bits 0-7 of the label number will be put into the blue channel, bits 8-15 into the green channel, and bits 16-23 into the red channel).

Stacks and tiled stacks:	.PNG, .TIF uncompressed, .RAW
3D Volume files:	.NII (NIfTI) 8 bit only (currently unsupported)

Exporting of screen shots

Stacks and tiled stacks:	.PNG, .TIF uncompressed, .RAW (all 24 bit RGB)
3D Volume files:	.NII (Nifti) 8 bit only (currently unsupported)

Bibliography

[1] Cardona, A., Saalfeld, S., Preibisch, S., Schmid, B., Cheng, A., Pulokas, J., Tomancak, P. and Hartenstein, V.: *An Integrated Micro- and Macroarchitectural Analysis of the Drosophila Brain by Computer-Assisted Serial Section Electron Microscopy*, PLoS Biology, **8**(10), (2010), 1-17 (e1000502).

[2] Cardona A, Saalfeld S, Schindelin J, Arganda-Carreras I, Preibisch S, et al.: *TrakEM2 Software for Neural Circuit Reconstruction*, PLoS ONE, **7**(6):e38011, (2012), doi:10.1371/journal.pone.0038011.

[3] Peng, H., Ruan, Z., Long, F., Simpson, J.H. and Myers, E.W.: *V3D enables real-time 3D visualization and quantitative analysis of large-scale biological image data sets*, Nature Biotechnology, **28**(4), (2010), 348-353.

[4] Saalfeld, S., Fetter, R., Cardona, A., and Tomancak, P.: *Elastic volume reconstruction from series of ultra-thin microscopy sections*, Nature Methods, **9**(7), (2012), 717-720.